

Nom:

### Exercici 1

Copia a la dreta aquestes quatre tasques del pipeline gràfic, però ordenades d'acord amb l'ordre d'execució.

- Alpha Blending
- Fragment shader
- Geometry shader
- Rasterització

Geometry shader  
Rasterització  
Fragment shader  
Alpha Blending

### Exercici 2

Copia a la dreta aquestes quatre tasques del pipeline gràfic, però ordenades d'acord amb l'ordre d'execució.

- Depth test
- glDrawElements
- Stencil Test
- Vertex Shader

glDrawElements  
Vertex Shader  
~~Depth Test~~ Stencil Test  
~~Stencil Test~~ Depth Test

### Exercici 3

Escriu quin és l'espai de coordenades inicial i final de la multiplicació de la **modelViewProjectionInverse** per un vèrtex.

Inicial: **Clip space**

Final: **Object space**

#### Exercici 4

Escriu, per cada tasca, en quin o quins shaders (VS, GS, FS) és possible:

- (a) discard FS
- (b) EndPrimitive() GS
- (c) Escriure gl\_Position VS + GS
- (d) dFdx, dFdy FS

#### Exercici 5

Què coordenada del fragment modifica la funció **glPolygonOffset**?

Z

En quin espai la modifica?

Window space

#### Exercici 6

Indica, amb la notació vista a classe, el light path que explica el color dominant dels píxels indicats a la imatge.

LDSE



#### Exercici 7

Indica, per cada punt, si pot ser dins (DINS) o segur que no (FORA) la piràmide de visió d'una càmera perspectiva:

- (a) (0.2, -1.5, 1.8, 2) en clip space DINS (tot entre +/- w)
- (b) (0, 0, 1, 1) en eye space FORA (z no negativa)
- (c) (200, 300, 400) en world space DINS (pot ser...)
- (d) (0, 0, 1, 1) en object space DINS (pot ser...)

### Exercici 8

Re-escriu el codi GLSL subratllat amb una versió més compacte:

```
float t;  
vec3 color1, color2;  
vec3 color = t*color1 + (1-t)*color2;
```

**vec3 color = mix(color2, color1, t);**

### Exercici 9

Re-escriu aquest codi GLSL amb una versió més compacte:

```
vec3 obs = (modelViewMatrixInverse * vec4(0,0,0,1)).xyz;
```

**vec3 obs = modelViewMatrixInverse[3].xyz;**

### Exercici 10

Escriu un exemple d'algorisme que suporti els light paths que s'indiquen:

(a)  $LS * DS * E$  (i  $LS * E$ ) **Two-pass raytracing**

(b)  $L(D|S) * E$  **Path tracing**

### Exercici 11



Volem aplicar la textura a un quad que té coordenades de textura inicials en [0,1].



Completa el FS per aconseguir el resultat que es mostra:

```
frontColor = texture(colorMap, _____ * vtexCoord);
```

**vec2(1,0.5)\***

## Exercici 12

Aquest VS calcula coordenades de textura projectives per a un FS que implementa shadow mapping:

```
uniform mat4 lightMatrix, modelMatrix;  
out vec4 textureCoords;  
...  
void main() {  
    ...  
    textureCoords = lightMatrix*modelMatrix*vec4(vertex,1);  
    gl_Position = modelViewProjectionMatrix *vec4(vertex,1);  
}
```

Usant aquesta notació:

$S(s_x, s_y, s_z)$  -> Scale matrix

$T(t_x, t_y, t_z)$  -> Translate matrix

$M$  -> model matrix (of the object)

$V$  -> view matrix (of the light camera)

$P$  -> projection matrix (of the light camera)

Escriu (com a producte de matrius) com l'aplicació ha de calcular la matriu pel uniform **lightMatrix**.

$T(0.5)S(0.5)PV$

## Exercici 13

A l'equació general del rendering:

$$L_o(\mathbf{x}, \omega_o, \lambda, t) = L_e(\mathbf{x}, \omega_o, \lambda, t) + \int_{\Omega} f_r(\mathbf{x}, \omega_i, \omega_o, \lambda, t) L_i(\mathbf{x}, \omega_i, \lambda, t) (\omega_i \cdot \mathbf{n}) d\omega_i$$

Què representa  $\Omega$ ?

Totes les direccions en una semi-esfera centrada al punt.

## Exercici 14

Escriu la matriu o producte de matrius per convertir un vèrtex de *object space* a *eye space*, usant únicament les matrius que s'indiquen (no en falta cap per aquest exercici):

modelMatrix	modelMatrixInverse
projectionMatrix	projectionMatrixInverse
modelViewProjectionMatrix	modelViewProjectionMatrixInverse

$\text{projectionMatrixInverse} * \text{modelViewProjectionMatrix}$

### Exercici 15

Indica quin tipus GLSL (float, vec2, vec3...) usaries per cada cas:

- (a) Segon paràmetre d'una crida a texture(colormap...) vec2
- (b) Coordenades de textura que passa VS a FS en shadow mapping vec4
- (c) Coordenades de textura que passa VS a FS en projective texture mapping vec4
- (d) Vector per accedir a un sphere map vec3

### Exercici 16

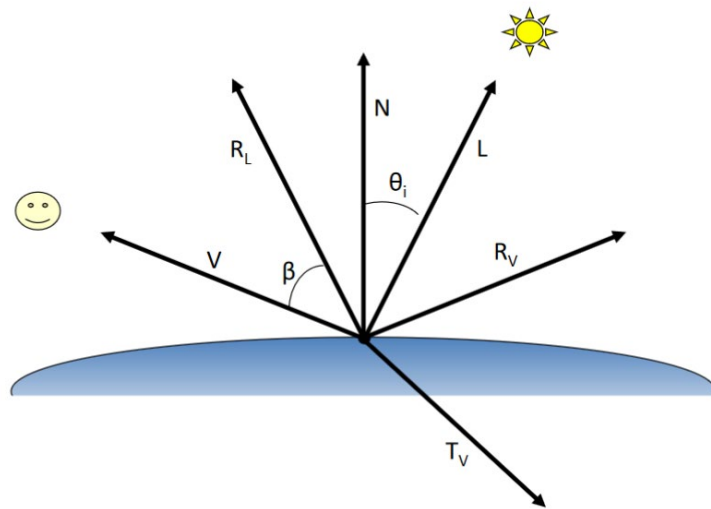
Amb la notació de la figura, indica, en el cas de Ray-tracing

- (a) Quin vector té la direcció del *shadow ray*?

L

- (b) Quin vector és paral·lel al raig transmès?

T



### Exercici 17

Continuant amb la figura anterior...

- (a) Quin vector té la direcció del raig primari?

V

- (b) Quins dos vectors determinen la contribució de Phong?

RL i V

### Exercici 18

Quines són les unitats de la radiància (radiometria) en el Sistema Internacional?

$W/(m^2 * sr)$

### Exercici 19

Completa, a sota, el codi que falta.

```
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
            
        fsi
    sino color:=colorDeFons
    fsi
sino color:=Color(0,0,0); // o colorDeFons
fsi
retorna color
ffunció
```

### Exercici 20

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

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

void main()
{
    

    float NdotL = max(0.0, dot(N,L));
    vec4 color = vec4(NdotL);
    vec2 st = vtxCoord.st / vtxCoord.q;

    float trueDepth = vtxCoord.p / vtxCoord.q;
    if (trueDepth <= storedDepth) fragColor = color;
    else fragColor = vec4(0);
}
```

---

Nom i Cognoms:

---

**Exercici 1**

Aquesta funció implementa el típic model d'il·luminació (Lambert+Phong):

```
vec4 light(vec3 N, vec3 V, vec3 L)
{
    V=normalize(V);
    L=normalize(L);
    vec3 R = normalize( 2.0*dot(N,L)*N-L );
    float NdotL = max( 0.0, dot( N,L ) );
    float RdotV = max( 0.0, dot( R,V ) );
    float Idiff = NdotL;
    float Ispec = 0;
    if (NdotL>0) Ispec=pow( RdotV, matShininess );
    return
        matAmbient * lightAmbient +
        matDiffuse * lightDiffuse * Idiff +
        matSpecular * lightSpecular * Ispec;
}
```

**Descriu clarament** la interpretació geomètrica dels paràmetres V i L de la funció.

= vector del punt cap a l'observador

L = vector del punt cap a la font de llum

**Exercici 2**

Indica quins tres vectors hauria de passar el VS al FS per que aquest últim pugui passar un vector qualsevol de tangent space a object space.

T, B, N (tangent, bitangent, normal).

### Exercici 3

Sigui  $P(u,v)$  la representació paramètrica d'una superfície. Sigui  $F(u,v)$  un mapa d'elevacions. Escriu l'expressió que permet calcular la superfície  $P'(u,v)$  que resulta de pertorbar  $P$  segons el mapa d'elevacions, tal i com es fa servir en *displacement mapping*.

$$P'(u,v) = P(u,v) + F(u,v) \cdot \frac{N(u,v)}{\|N(u,v)\|}$$

### Exercici 4

Aquesta equació relaciona, per bump mapping, la normal de la superfície pertorbada amb la normal original:

$$N' = \frac{\partial P}{\partial u} \times \frac{\partial P}{\partial v} + \frac{\frac{\partial F}{\partial u} \left( N \times \frac{\partial P}{\partial v} \right)}{\|N\|} + \frac{\frac{\partial F}{\partial v} \left( \frac{\partial P}{\partial u} \times N \right)}{\|N\|} + \frac{\frac{\partial F}{\partial u} \frac{\partial F}{\partial v} (N \times N)}{\|N\|^2}$$

(a) Indica, dels quatre termes, quin avalua a un vector nul. Justifica la resposta.

El darrer terme, ja que  $N \times N = \mathbf{0}$

(b) Encercla a l'equació anterior un parell de vectors que siguin tangents a la superfície original.

Els vectors de la forma  $N \times \dots$  ó  $\dots \times N$

### Exercici 5

Tenim un bump map on cada texel conté les dues components ( $du$ ,  $dv$ ) del gradient del camp d'elevacions  $F(u,v)$ . Indica clarament (ex. codi GLSL) com podem calcular les components de la normal pertorbada  $N'$ , en *espai tangent*, a partir de ( $du, dv$ ).

`vec3 N' = normalize(-du, -dv, 1)`

### Exercici 6

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

- Crida a `dFdx`
- Declaració dels atributs vèrtex, normal, `texCoord...`
- Depth test
- Rasterització

Declaració dels atributs (VS)

Rasterització

Crida a `ddFdx` (FS)

Depth test



### Exercici 7

Indica i declara en GLSL 3.30 core els dos vec3 cal que el VS passi al FS per tal d'usar el model de Phong al FS.

Al VS:

```
out vec3 vnorm; //
```

```
out vec3 pos; // (vec4 també és correcte))
```

Al FS:

```
in vec3 vnorm;
```

```
in vec3 pos; // (vec4 també és correcte))
```

### Exercici 8

Completa aquest FS de forma que, amb la textura de l'esquerra, produeixi la imatge de la dreta, quan s'aplica a l'objecte plane.obj, que té coordenades de textura del (0,0) al (1,1).



```
void main() {  
    vec2 f = vec2(....., .....);  
    fragColor = texture(colormap, f * vtexcoord);  
}  
  
vec2((-3, 1)
```

### Exercici 9

Aquest fragment de codi està generant coordenades de textura a partir de dos plans S, T: `vec2`

```
st = vec2(dot(vec4(vertex,1.0), planeS), dot(vec4(vertex,1.0), planeT));
```

Si tenim aquests uniforms,

```
uniform vec4 planeS = vec4( 4, 0, 0,  
0 ); uniform vec4 planeT = vec4( 0, 1,  
0, 0 );
```

Quines coordenades de textura (s,t) es calcularan pel vèrtex (2, 12, 127, 1)?

->  $(4 \cdot 2, 1 \cdot 12) = (8, 12)$

### Exercici 10

Escriu en codi GLSL com calcular la posició de la càmera en *object space*. Es valorarà la eficiència.

```
vec3 obs = modelViewMatrixInverse[3].xyz;
```

Més eficient que `(modelViewMatrixInverse*vec4(0,0,0,1)).xyz;`

### Exercici 11

Indica clarament una tècnica utilitzada per calcular coordenades de textura per la qual **no** es pot assumir que les coordenades de textura (s,t) s'interpolen linealment en espai objecte. Explica la raó d'aquesta no linearitat.

Projective texture mapping. El motiu és que les coordenades de textura de cada punt es veuen afectades per la deformació de perspectiva associada al "projector".

### Exercici 12

Indica en quin interval (el més ajustat possible) poden estar les coordenades Z dels punts interiors a la piràmide de visió d'una càmera perspectiva, segons l'espai considerat. Pots fer servir  $\pm\infty$  si s'escau.

- Object space:  $(-\infty, +\infty)$
- Eye space:  $(-\infty, 0)$
- NDC:  $[-1, 1]$
- Window space:  $[0, 1]$

### Exercici 13

En molts shaders sovint ens trobem amb la necessitat de transformar valors linealment d'un cert interval inicial a un cert interval de sortida. Per exemple, ens pot interessar transformar valors de l'interval  $[-1,1]$  a valors en l'interval  $[0,1]$ .

Escriu una funció `map()` en GLSL que, donat un valor `x` dins l'interval  $[imin,imax]$ , calculi el resultat interpolant linealment dins l'interval  $[omin, omax]$ , de forma que al valor `imin` li correspongui el valor `omin`, i al valor `imax` li correspongui `omax`.

```
float map(float x, float imin, float imax, float omin, float omax)
{
    float n = (x-imin)/(imax-imin); // n dins [0,1]
    return omin + n*(omax-omin); // dins [omin, omax]
}
```

### Exercici 14

Indica, per cadascun d'aquests modes de filtrat per MIPMAP, quants texels es fan servir per avaluar cada crida a `texture()`.

<code>GL_NEAREST_MIPMAP_NEAREST</code>	1
<code>GL_LINEAR_MIPMAP_NEAREST</code>	4
<code>GL_NEAREST_MIPMAP_LINEAR</code>	
<code>GL_LINEAR_MIPMAP_LINEAR</code>	822

### Exercici 15

Observa la següent figura. Indica, per cada tècnica, si l'ha pogut reproduir o no, **justificant cada resposta**.

- (a) Bump mapping      No, s'observen oclusions
- (b) Parallax mapping:      No, s'observen oclusions
- (c) Relief mapping:      Si, es plausible
- (d) Displacement mapping:      Si, es plausible



### Exercici 16

Per una determinada primitiva, sabem les coordenades de textura que tindran alguns fragments:

Coordenades (x,y)	Coordenades (s,t)
-------------------	-------------------

100.5, 100.5	0.3, 0.1
--------------	----------

101.5, 100.5	0.5, 0.8
--------------	----------

Indica quin serà el resultat d'avaluar

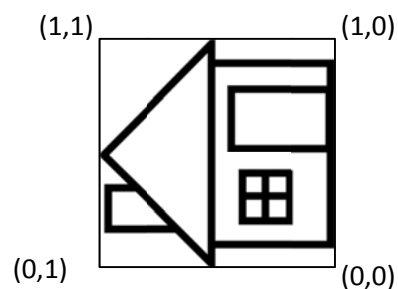
$dFdx(texCoord)$

per aquests fragments.

$: (0.5, 0.8) - (0.2, 0.7) \quad (\rightarrow (0.3, 0.1)$

### Exercici 17

Amb la textura de l'esquerra hem texturat el quad de la dreta. Indica, al mateix quad, les coordenades de textura (s,t) de cada vèrtex.



Quin valor tindran  $dFdy(texCoord.s)$  i  $dFdx(texCoord.t)$  als fragments del quad anterior?

$dFdy(texCoord.s) = dFdx(texCoord.t)$

**Exercici 18**  $CCoord.t) = 0$ .

Escriu codi en llenguatge GLSL 3.30 core per passar un punt P de clip space a model space

```
vec4 P;
```

```
...
```

```
P = modelViewProjectionMatrixInverse * P;
```

---

Nom i Cognoms:

---

Tots els exercicis tenen el mateix pes.

**Exercici 1**

Aquí teniu una llista d'etapes/tasques del pipeline gràfic, ordenades per ordre alfabètic. Torna-les a escriure a la dreta, però ordenades segons l'ordre al pipeline gràfic. Suposa que només hi ha un VS i un FS (no hi ha GS).

- Alpha blending
- Crides a dFdx, dFdy
- Pas de coordenades a clip space
- Rasterització

Pas de coordenades a clip space

Rasterització

Crides a dFdx, dFdy

Alpha blending

**Exercici 2**

Hem produït un fragment amb color RGBA = (1.0, 0.5, 0.0, 0.4) corresponent al pixel (i,j). El color RGBA del pixel (i,j) al buffer de color és (0.5, 0.5, 1.0, 1.0). Si tenim activat alpha blending amb la crida

`glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)`

quin serà el color resultant al buffer de color (assumint que passa tots els tests)?

$$0.4 * 1.0 + 0.6 * 0.5 = 0.7$$

$$0.4 * 0.5 + 0.6 * 0.5 = 0.5$$

$$0.4 * 0.0 + 0.6 * 1.0 = 0.6$$

$$0.4 * 0.4 + 0.6 * 1.0 = 0.76 \rightarrow (0.7, 0.5, 0.6, 0.76)$$

**Exercici 3**

Indica quina és la matriu que aconseguix la conversió demanada (assumint que no hi ha transformació de modelat), usant aquestes abreviatures:

MV = `gl_ModelViewMatrix`

MVP = `gl_ModelViewProjectionMatrix`

$MV^{-1}$  = `gl_ModelViewMatrixInverse`

$MVP^{-1}$  = `gl_ModelViewProjectionMatrixInverse`

NM = `gl_NormalMatrix`

a) Pas d'un vèrtex de object space a eye space    MV

b) Pas d'un vèrtex de eye space a world space     $MV^{-1}$

c) Pas d'un vèrtex de clip space a object space     $MVP^{-1}$

d) Pas de la normal de object space a eye space    NM

#### Exercici 4

En una hipotètica versió de GLSL, no està definit el uniform **gl\_ModelViewProjectionMatrix** (però sí la resta de matrius de GLSL). Escriu, en codi GLSL vàlid, com faries la conversió de **gl\_Vertex** d'object space a clip space (assumint que no hi ha transformació de modelat).

```
gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
```

->

```
gl_Position = gl_ProjectionMatrix * gl_ModelViewMatrix * gl_Vertex;
```

#### Exercici 5

Completa el següent FS per tal que calculi correctament la contribució difosa:

```
varying vec3 normal; // eye space
```

```
varying vec3 pos; // pos en eye space
```

```
vec4 lightSource( vec3 N, vec3 V, gl_LightSourceParameters light ) {
```

```
    vec3 L = normalize( light.position.xyz - V );
```

```
    vec3 R = normalize( 2.0*dot(N,L)*N-L );
```

```
    V = normalize (-V.xyz);
```

```
    float diff; // cal que el calculeu vosaltres
```

```
    diff = max( 0.0, dot( N,L ) );
```

```
return gl_FrontMaterial.diffuse * light.diffuse * diff;  
}
```

#### Exercici 6

Per què ens pot ser útil aquesta sentència GLSL?

```
vec3 foo = gl_ModelViewMatrixInverse[3].xyz;
```

--> Per obtenir la posició del observador, en object space (o world space, si no hi ha transf de modelat).

#### Exercici 7

Una textura de 1024x1024 requereix emmagatzemar en memòria, òbviament, 1M texel. Quans texels cal emmagatzemar si la textura fa servir mipmapping?

$1024 \times 1024 + 512 \times 512 + 256 \times 256 + \dots + 1 \times 1 = \sum_{i=0}^{10} 2^{2i} = 1.33 \text{ M texel}$

## Exercici 8

Aquest pseudocodi implementa environment mapping:

1. Calcular el vector unitari X
2. Calcular el vector R com a reflexió de X respecte la normal al punt
3. Utilitzar R per accedir al environment map i obtenir el color de l'entorn en direcció R

Què és el vector X del pas 1, i com el podeu calcular?

Vector L: vector unitari del punt cap a l'observador

## Exercici 9

Quina magnitud de radiometria (o fotometria) representa millor l'energia que arriba a cada diferencial de superfície, i que es calcula al primer pass (light pass) de la tècnica two-pass raytracing?

Irradiancia

Indica també una possible unitat de mesura d'aquesta magnitud.

(W/m<sup>2</sup>, lux)

## Exercici 10

Fes matching entre les magnituds/conceptes de radiometria de l'esquerra, i els conceptes de la dreta:

Flux	Energia d'un raig de llum
Radiància	Quantitat total d'energia emesa per una font de llum (per unitat de temps)
Intensitat BRDF	Propietats de reflectivitat d'un material
Flux	Quantitat total d'energia emesa per una font de llum (per unitat de temps i angle sòlid)
Radiància	Quantitat total d'energia emesa per una font de llum (per unitat de temps)
Intensitat	Energia d'un raig de llum
BRDF	Quantitat total d'energia emesa per una font de llum (per unitat de temps i angle sòlid)
	Propietats de reflectivitat d'un material

## Exercici 11

Per un determinat objecte, tenim emmagatzemada la següent informació, relativa a la seva transformació de modelat: centre de l'objecte, vector de translació, matriu de rotació, paràmetres de l'escalat. Indica, **com a producte de 4 ó 5 matrius**, com calcular la transformació de modelat de l'objecte (vigileu l'ordre!). Feu servir la notació:

T(x,y,z) -> matriu de translació segons el vector (x,y,z)

R -> matriu de rotació emmagatzemada

S(x,y,z) -> matriu d'escalat segons els factors d'escalat x,y,z

T(centre)\*T(trans)\*R\*S\*T(-centre)

## Exercici 12

Quin és el principal avantatge d'utilitzar VBO en comptes de VA?

Bàsicament un VBO és un VA emmagatzemat en memòria del servidor OpenGL (ex. GPU).

Per tant, estalvi en les dades que han de passar-se cada frame de RAM a memòria de la GPU.

## Exercici 13

Tenim un normal map on les components RGB de cada texel codifiquen les components  $(nx', ny', nz')$  en espai tangent d'un vector unitari en la direcció de la normal pertorbada. Donats els vectors T, B, N (tangent, bitangent i normal) d'una base ortonormal, en eye space, corresponents a un fragment, indica com calcularies la normal pertorbada  $N'$  en eye space.

Simplement hem de passar la normal  $(nx', ny', nz')$  de tangent space a eye space:  $[T \ B \ N] * (nx', ny', nz')$  on  $[T \ B \ N]$  és una matriu 3x3 que té per columnes els tres vectors T, B, N.

## Exercici 14

Quins són els sistemes de coordenades origen i destí de la transformació de projecció (projection transform), representada per la `gl_ProjectionMatrix` en GLSL?

Eye Space --> Clip Space

## Exercici 15

Indica quina condició han de satisfer les coordenades  $(x, y, z, w)$  d'un vèrtex en clip space, per tal de ser interior al frustum de visió d'una càmera perspectiva.

$-w \leq x \leq w$  (el mateix per  $y, z$ )

## Exercici 16

Aquesta és la declaració de la funció OpenGL `glColorMask()`:

```
void glColorMask( GLboolean red, GLboolean green, GLboolean blue, GLboolean alpha )
```

a) Quin efecte té aquesta funció?

Activar (true) / desactivar (false) l'escriptura en els buffers de color.

b) Menciona una tècnica concreta (dintre de les tècniques per simular ombres, reflexions, transparències...) en la qual es faci servir, i perquè.

Ombres per projecció amb stencil, als passos on només volem modificar el depth buffer o el stencil buffer, però no pas el color buffer.



### Exercici 17

Completa (amb els uniforms apropiats) aquestes en línies en codi GLSL vàlid (assumint que només es pinten les cares frontface):

a) float shininess = `gl_FrontMaterial.shininess` // exponent de reflectivitat especular del material

float shininess = `gl_FrontMaterial.shininess`

b) vec4 posLlum = `gl_LightSource[0].position` // posició en eye space de la primera llum

vec4 posLlum = `gl_LightSource[0].position`

### Exercici 18

De quin tipus GLSL (float, vec2, vec3...) és el resultat d'aquestes

expressions? a) `dot(vec3(1,0,0), vec3(0,1,0))`

float

a) `texture2d(sampler, vec2(0.5,0.5));`

vec4

b) `cross (vec3(1,0,0,), vec3(0, 1, 0))`

vec3

c) `vec3(1,0,0) * vec3(0, 1, 0)`

vec3

### Exercici 19

Què està fent aquest codi i en quina tècnica s'utilitza?

```
glLoadIdentity();  
glTranslated(0.5,  
0.5,0.5); glScaled(0.5,  
0.5, 0.5);  
gluPerspective(...);  
gluLookAt(...)
```

Està definint la matriu de conversió world space -> texture space; es fa servir a proj texture mapping.

### Exercici 20

En una aplicació volem reproduir l'ombra que projecten diferents globus aerostàtics sobre una pista plana i horitzontal, al vespre (sol prop de la posta), des de diferents punts de vista sobre la pista. Quin inconvenient pot tenir usar shadow mapping en aquest context?

Aliasing greu a l'ombra degut al problema "duelling frustra".

# Preguntes per a l'avaluació de les competències transversals

## Pregunta 1

A què fa referència el terme *haptic*?

- (a) Interacció en aplicacions mèdiques
- (b) Interacció explícita
- (c) Manipulació d'objectes deformables
- (d) Realimentació tàctil

## Pregunta 2

Indica al menys un parell de tècniques d'adquisició de dades de volum en medicina.

CT, MR, ultrasons...

---

Nom i Cognoms:

---

Tots els exercicis tenen el mateix pes.

**Exercici 1**

Aquí teniu una llista d'etapes/tasques del pipeline gràfic, ordenades per ordre alfabètic. Torna-les a escriure a la dreta, però ordenades segons l'ordre al pipeline gràfic. Suposa que només hi ha un VS i un FS (no hi ha GS).

- Escritura de `gl_Position`
- Rasterització
- Retallat de primitives (clipping)
- Stencil test

<p>Escritura de <code>gl_Position</code></p> <p>Retallat de primitives (clipping)</p> <p>Rasterització</p> <p>Stencil test</p>
--------------------------------------------------------------------------------------------------------------------------------

**Exercici 2**

Indica, per cada etapa de la llista de sota, si és ANTERIOR o POSTERIOR a la etapa de rasterització:

- |                                                              |           |
|--------------------------------------------------------------|-----------|
| (a) Primitive assembly                                       | ANTERIOR  |
| (b) Processament del fragment                                | POSTERIOR |
| (c) Ús de les funcions <code>dFdx</code> , <code>dFdy</code> | POSTERIOR |
| (d) Pas de les coordenades a clip space                      | ANTERIOR  |

**Exercici 3**

Indica al menys una tasca típica al pipeline de visualització programable, que es pugui fer tant abans com després de la rasterització.

Càlculs d'il·luminació; generació de coordenades de textura (sphere mapping...), ...

#### Exercici 4

Hem produït un fragment amb color RGBA = (1.0, 0.5, 0.0, 0.8) corresponent al pixel (i,j). El color RGBA del pixel (i,j) al buffer de color és (1.0, 0.5, 1.0, 1.0). Si tenim activat alpha blending amb la crida

```
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)
```

quin serà el color RGBA resultant al buffer de color (assumint que passa tots els tests)?

$$0.8 * 1.0 + 0.2 * 1.0 = 1.0$$

$$0.8 * 0.5 + 0.2 * 0.5 = 0.5$$

$$0.8 * 0.0 + 0.2 * 1.0 = 0.2$$

$$0.8 * 0.8 + 0.2 * 1.0 = 0.84 \rightarrow (1.0, 0.5, 0.2, 0.84)$$

#### Exercici 5

Indica quina és la matriu que aconseguix la conversió demanada (assumint que no hi ha transformació de modelat), usant aquestes abreviatures:

MV = `gl_ModelViewMatrix`

MVP = `gl_ModelViewProjectionMatrix`

$MV^{-1}$  = `gl_ModelViewMatrixInverse`

$MVP^{-1}$  = `gl_ModelViewProjectionMatrixInverse`

NM = `gl_NormalMatrix`

I = Identitat

a) Pas d'un vèrtex de eye space a world space  $MV^{-1}$

b) Pas d'un vèrtex de clip space a object space  $MVP^{-1}$

c) Pas de la normal de object space a eye space NM

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

#### Exercici 6

En una hipotètica versió de GLSL, no està definit el uniform **`gl_ModelViewProjectionMatrixInverse`** (però sí la resta de matrius de GLSL). Escribeu, en codi GLSL vàlid, com faríeu la conversió de **`vec4 P`** de clip space a object space (assumint que no hi ha transformació de modelat).

```
gl_ModelViewProjectionMatrixInverse * P;
```

->

```
gl_ModelViewMatrixInverse * gl_ProjectionMatrixInverse * P;
```

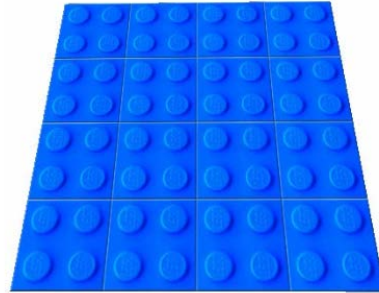
### Exercici 7

Completa el següent FS per tal que calculi correctament el vector L que intervé a la contribució

```
difosa: vec4 lightSource( vec3 N, vec3 P, gl_LightSourceParameters light ) {  
    // N és la normal en eye  
space // P és el punt en eye  
space  
vec3 L = normalize( light.position.xyz - P );  
  
    float diff = max( 0.0, dot( N,L ) );  
return gl_FrontMaterial.diffuse * light.diffuse *  
diff;
```

### Exercici 8

Amb la textura de l'esquerra, volem texturar l'objecte Plane com a la dreta.



Completa la línia que necessitem al VS:

```
gl_TexCoord[0].st = 4.0 * gl_MultiTexCoord0.st;
```

### Exercici 9

Quants nivells de detall (nivells de LOD) formen la piràmide mipmapping completa d'una textura de 256x256 texels (tenint en compte el LOD 0)?

$1 + \log_2(256) = 9$  (256x256, 128x128, 64x64, 32x32, 16x16, 8x8, 4x4, 2x2, 1x1)

### Exercici 10

La següent figura mostra la textura d'un tauler d'escacs (a l'esquerra) i el Plane texturat (a la dreta):



El vèrtex inferior esquerra del polígon té coordenades de textura (0,0), i el vèrtex superior dret (1,1). Indica, a la imatge de la dreta, en quina regió es maximitza el valor de  $\frac{\partial s}{\partial x}$

Es maximitza a l'aresta superior del polígon.

### Exercici 11

Tenim un model amb **coordenades de textura 2D que cobreixen l'interval [-1,1]**. Ens demanen un VS que mostri el model projectat sobre aquest espai de textura (com a l'exercici UVunfold), de forma que ocupi tot el viewport, amb independència de la càmera. Completa aquesta línia del VS demanat:

```
gl_Position = vec4( gl_TexCoord[0].s, gl_TexCoord[0].t, 0.0, 1.0);
```

### Exercici 12

Imagina un FS al qual li arriben les coordenades del punt (**vec4 P**) en un cert espai de coordenades.

Indica quin test hauries de fer sobre **P** per tal **d'eliminar (discard) els fragments que cauen a la meitat dreta** de la finestra OpenGL, amb mida 800x600:

a) Si **P** està en clip space

```
if ( P.x/P.w > 0 ) discard;
```

b) Si **P** està en NDC

```
if ( P.x > 0 ) discard;
```

c) Si **P** està en window space

```
if ( P.x > 400 ) discard;
```

### Exercici 13

Escriu quina matriu 4x4 necessitem per simular la reflexió de l'escena respecte un mirall situat al pla

```
Y=0.1  0  0  0
        0 -1  0  0
        0  0  1  0
        0  0  0  1
```

#### Exercici 14

Tenim un cub format per 6 cares i 8 vèrtexs.

a) Si volem que cada corner (vèrtex associat a una única cara) tingui la normal de la cara a la que pertany, quants vèrtexs necessitem en el VBO del cub?

6 cares \* 4 vèrtexs per cada cara = **24 vèrtexs**

b) Si volem que cada vèrtex del cub tingui com a normal el promig de les normals de les cares que incideixen al vèrtex, quants vèrtexs necessitem (com a mínim) en el VBO del cub?

**8 vèrtexs (mínim)**

#### Exercici 15

Tenim un normal map on les components RGB de cada texel codifiquen les components ( $n_x', n_y', n_z'$ ) en espai tangent d'un vector unitari en la direcció de la normal pertorbada. Donats els vectors T,B,N (tangent, bitangent i normal) d'una base ortonormal, en eye space, corresponents a un fragment, indica com calcularies la normal pertorbada  $N'$  en eye space.

Simplement hem de passar la normal ( $n_x', n_y', n_z'$ ) de tangent space a eye space:  $[T \ B \ N] * (n_x', n_y', n_z')$  on  $[T \ B \ N]$  és una matriu 3x3 que té per columnes els tres vectors T, B, N.

#### Exercici 16

Quins són els sistemes de coordenades origen i destí de la transformació representada per la `gl_ModelViewMatrix` en GLSL?

Object Space --> Eye Space

#### Exercici 17

Aquesta és la declaració de la funció OpenGL `glDepthMask()`:

```
void glDepthMask( GLboolean foo)
```

Quin efecte té aquesta funció?

Activar (true) / desactivar (false) l'escriptura en el depth buffer.

#### Exercici 18

De quin tipus GLSL (float, vec2, vec3...) és el resultat d'aquestes expressions?

a) `dot(vec3(1,0,0), vec3(0,1,0))`

float

b) `cross (vec3(1,0,0,), vec3(0, 1, 0))`

vec3

### Exercici 19

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

- a) Shadow mapping, suposant que la llum és dinàmica 2 cops
- b) Ombres per projecció, versió sense stencil 2 cops
- c) Reflexió amb objectes virtuals, amb stencil (ignoreu els passos en que només es dibuixa el mirall): 2 cops
- d) Projective Texture Mapping, amb una textura fixa 1 cop

### Exercici 20

Tenim una aplicació amb una escena opaca, sense miralls, amb una única font de llum puntual situada al punt (0,0,0) en coordenades de l'observador. Quina tècnica creus apropiada per reproduir les ombres en aquest cas?

Cap; en aquest cas, la càmera coincideix amb la llum, i per tant les ombres no seran visibles.

CT, MR, ultrasons...



---

Nom i Cognoms:

---

Tots els exercicis tenen el mateix pes.

**Exercici 1**

Aquí teniu una llista d'etapes/tasques del pipeline gràfic, ordenades per ordre alfabètic. Torna-les a escriure a la dreta, però ordenades segons l'ordre al pipeline gràfic. Suposa que només hi ha un VS i un FS (no hi ha GS).

- Divisió de perspectiva
- Escriptura de `gl_Position`
- `glDrawElements`
- Rasterització

`glDrawElements`

Escriptura de `gl_Position`

Divisió de perspectiva

Rasterització

**Exercici 2**

Indica, per cada tasca/etapa de la llista de sota, si és ANTERIOR o POSTERIOR a la etapa de rasterització:

- |                     |           |
|---------------------|-----------|
| (a) Geometry Shader | ANTERIOR  |
| (b) Fragment Shader | POSTERIOR |
| (c) Stencil Test    | POSTERIOR |
| (d) Alpha blending  | POSTERIOR |

**Exercici 3**

Quants nivells de detall (nivells de LOD) formen la piràmide mipmaping completa d'una textura de 512x512 texels (tenint en compte el LOD 0)?

$$1 + \log_2(512) = 10 \quad (512 \times 512, 256 \times 256, 128 \times 128, 64 \times 64, 32 \times 32, 16 \times 16, 8 \times 8, 4 \times 4, 2 \times 2, 1 \times 1)$$

#### Exercici 4

Tenim un octaedre representat amb una malla triangular formada per 6 vèrtexs i 8 triangles. Volem construir un VBO per representar aquest octaedre, 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 l'octaedre aparegui il·luminat correctament). Cada coordenada/component estarà representada per un float (4 bytes).

(a) Quants vèrtexs necessitem representar al VBO?

$8 \text{ tri} * 3 \text{ v/tri} = \mathbf{24 \text{ vèrtexs}}$  (hem de repetir vèrtexs perquè no comparteixen normals)

(b) Quants índexs (*elements*) es necessiten a l'array d'índexs?

$8 \text{ tri} * 3 \text{ índexs/tri} = \mathbf{24 \text{ índexs}}$

(c) Quina mida (en bytes) tindrà l'array de coordenades de vèrtexs?

$24 \text{ v} * 3 \text{ coord/v} * 4 \text{ bytes/coord} = \mathbf{288 \text{ bytes}}$

#### Exercicis 5 i 6

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

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

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

b) Pas d'un vèrtex de object space a eye space       $V * M$

c) Pas d'un vèrtex de eye space a clip space      P

d) Pas d'un vèrtex de eye space a world space       $V^{-1}$

A) Pas d'un vèrtex de clip space a object space       $M^{-1} * V^{-1} * P^{-1}$

B) Pas d'un vèrtex de world space a clip space       $P * M$

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

D) Pas de la normal de object space a eye space      N

### Exercici 7

Tenim activat alpha blending amb la crida

```
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)
```

Hem produït un fragment amb color RGBA = (1.0, 0.5, 0.0, **a**) corresponent al pixel (i,j). El color RGBA del pixel (i,j) al buffer de color és (1.0, 1.0, 0.0, 0.0). Quin valor ha de tenir **a** si volem que les components RGB del resultat siguin (1.0, 0.8, 0.0)?

D'acord amb la component G:  $a \cdot 0.5 + (1-a) \cdot 1.0 = 0.8 \rightarrow \mathbf{a=0.4}$

### Exercici 8

Indica quina és la diferència més important entre els models d'il·luminació local i els models d'il·luminació global.

Local  $\rightarrow$  només llum directa

Global  $\rightarrow$  llum directa + llum indirecta

### Exercici 9

Completa el següent FS per tal que calculi correctament el vector L que intervé a la contribució difosa:

```
uniform vec4 matDiffuse;
```

```
uniform vec4 lightDiffuse, lightPosition;
```

```
vec4 lightSource( vec3 N, vec3 P) {
```

```
    // N és la normal en eye space    // P és el punt en eye space
```

```
    vec3 L = normalize( lightPosition.xyz - P );
```

```
    float diff = max( 0.0, dot( N,L ) );
```

```
    return matDiffuse * lightDiffuse *
```

```
    diff;
```

### Exercici 10

Indica, en la notació estudiada a classe,  $L(D|S)*E$ , quins light paths són suportats per ray-tracing

clàssic.  $LDS*E$  i  $LS*E$

### Exercici 11

Volem generar amb RayTracing una imatge 1024x1024 d'una escena interior tancada. Tots els objectes són opacs i perfectament difosos, i hi ha quatre fonts de llum.

- a) Quants rajos primaris caldrà traçar?  $1024 \times 1024 = 2^{20}$  rajos.
- b) Quants shadow rays caldrà traçar, en total?  $2^{20}$  hits \* 4 shadow rays/hit =  $2^{22}$  shadow rays
- c) Quants rajos reflectits caldrà traçar, en total? Cap
- d) Quants rajos transmesos caldrà traçar, en total? Cap

### Exercici 12

Relaciona cada concepte de l'esquerra amb un concepte de la dreta:

- |                |                                                                                   |
|----------------|-----------------------------------------------------------------------------------|
| 1. Flux        | A. $\phi$ per unitat d'angle sòlid                                                |
| 2. Intensitat  | B. Mesurat en lúmens                                                              |
| 3. Irradiància | C. Energia (per unitat de temps) que travessa un punt en una determinada direcció |
| 4. Radiància   | D. Mesurat en lux                                                                 |

1  $\rightarrow$  B

2  $\rightarrow$  A

3  $\rightarrow$  D

4  $\rightarrow$  C

### Exercici 13

Completa l'equació general del rendering:

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int f(x, \omega_i, \omega_o) L_i(x, \omega_i) \cos(\theta_i) d\omega_i$$

### Exercici 14

Quin és el nom de l'equació que ens permet saber la quantitat de llum reflectida i la quantitat de llum transmesa, en funció de diferents paràmetres (angle incident, etc)?

Equacions de Fresnel

### Exercici 15

Volem dibuixar una escena amb alguns objectes opacs i altres translúcids, amb alpha blending. Donat que no volem ordenar els objectes, fem servir aquest codi:

```
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

if (depthTestOpaque) glEnable(GL_DEPTH_TEST);
else glDisable(GL_DEPTH_TEST);
glDepthMask(maskOpaque);
drawObjects(opaqueObjects);

if (depthTestTranslucent) glEnable(GL_DEPTH_TEST);
else glDisable(GL_DEPTH_TEST);
glDepthMask(maskTranslucent);
drawObjects(translucentObjects);
```

Indica els valors més adients per les variables booleans:

- |                          |       |
|--------------------------|-------|
| (a) depthTestOpaque      | true  |
| (b) depthTestTranslucent | true  |
| (c) maskOpaque           | true  |
| (d) maskTranslucent      | false |

### Exercici 16

Tenim una escena amb diferents objectes, cadascú amb la seva transformació de modelat. Volem simular les reflexions en un mirall pla fent servir la tècnica de reflexió amb objectes virtuals. Hem calculat la matriu R de reflexió respecte el pla del mirall, usant els coeficients (a,b,c,d) del pla en world space. Fent servir la notació

M = modelMatrix (original)	$M^{-1}$ = modelMatrixInverse (original)
V = viewingMatrix	$V^{-1}$ = viewingMatrixInverse

indica quin producte de matrius és necessari **en el pas en que es dibuixen els objectes en posició virtual**:

- (a) Passar un vèrtex de object space a eye space

$$V * R * M$$

- (b) Passar un vèrtex de object space a world space

$$R * M$$

### Exercici 17

Tenim un model amb **coordenades de textura 2D que cobreixen l'interval [-1,1]**. Ens demanen un VS que mostri el model projectat sobre aquest espai de textura (com a l'exercici UVunfold), de forma que ocupi tot el viewport, amb independència de la càmera. Completa aquesta línia del VS demanat:

```
layout (location = 3) in vec2 texCoord;
```

```
gl_Position = vec4( texCoord.s, texCoord.t, 0.0, 1.0);
```

### Exercici 18

Tenim aquest fragment de

```
FS: void main() {  
    vec3 I = normalize(Pos);  
    vec3 R = reflect(I, N);  
    fragColor =  
    $sampleTexture(R);
```

(a) Quina tècnica està implementant?

Environment mapping

(b) En quin espai de coordenades està treballant?

Eye space; en cas contrari  $I = \text{normalize}(\text{Pos} - \text{Obs})$

### Exercici 19 i 20

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

a) Shadow mapping, suposant que la llum és dinàmica

2 cops

b) Ombres per projecció, versió sense stencil

2 cops

c) Reflexió amb objectes virtuals, amb stencil (ignoreu els passos en que només es dibuixa el mirall):

2 cops

d) Environment Mapping, amb una textura fixa

1 cop

## Preguntes per a l'avaluació de les competències transversals

### Pregunta 1

Explica què és i per què es fa servir la funció de transferència (transfer function) en aplicacions mèdiques

Proporciona el color i la opacitat a partir del valor del voxel.

### Pregunta 2

Explica per què serveix la tècnica de *Image Registration*, dins les aplicacions dels gràfics en medicina.

Per alinear (establir la correspondència espacial) els models de volum obtinguts amb les diferents modalitats de captació d'imatge mèdica (MRI, CT...)

## Gràfics Nom i Cognoms:

Tots els exercicis tenen el mateix pes.

**Exercici 1**

Aquí teniu una llista d'etapes/tasques del pipeline gràfic, ordenades per ordre alfabètic. Torna-les a escriure a la dreta, però ordenades segons l'ordre al pipeline gràfic. Suposa que només hi ha un VS i un FS (no hi ha GS).


- Divisió de perspectiva

- glDrawElements

- Rasterització

- Transformació a Clip Space

glDrawElements

Transformació a 

Space Divisió de

perspectiva Rasterització

**Exercici 2**

Indica, per cada tasca/etapa de la llista de sota, si és ANTERIOR o POSTERIOR a la etapa de

rasterització: (a) Geometry Shader

ANTERIOR

(b) Fragment



ader

POSTERIOR

(c)

dFdx,

dFdy

POSTERIOR

(d)

Stencil

Test

POSTERIOR

**Exercici 3**

El LOD 0 d'una textura té 1024 x 512 texels. Quina mida té el LOD 2 d'aquesta mateixa

textura?



LOD 0 → 1024 x 512 ; LOD 1 → 512 x 256; LOD 2 → **256 x 128**

#### Exercici 4

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).

(a) Quants vèrtexs necessitem representar al VBO?

$12 \text{ tri} * 3 \text{ v/tri} = 36 \text{ vèrtexs}$  (hem de repetir vèrtexs perquè no comparteixen normals)

(b) Quants índexs (*elements*) es necessiten a l'array d'índexs?

$12 \text{ tri} * 3 \text{ índexs} = 36 \text{ índexs}$

#### Exercicis 5 i 6

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

M = modelMatrix

$M^{-1}$  = modelMatrixInverse

= viewingMatrix

$V^{-1}$  = viewingMatrixInverse

P = projectionMatrix

$P^{-1}$  =

N = normalMatrix

projectionMatrixInverse

I = Identitat

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

P

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

$V^{-1}$

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

$V^{-1} * P^{-1}$

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

$P * V$

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

I

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

M

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

$V * M$

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

N



## Exercici 7

Tenim activat alpha blending amb la

crida `glBlendFunc(GL_ONE_MINUS_SRC_ALPHA, GL_SRC_ALPHA)`

Hem produït un fragment amb color RGBA = (1.0, 0.5, 0.0, 0.2) corresponent al pixel (i,j). El color RGBA del pixel (i,j) al buffer de color és (1.0, 1.0, 0.5, 0.0). Indica quin serà el color RGBA resultant del blending, amb les operacions que duen a aquest resultat.

$$(1-0.2) (1.0, 0.5, 0.0, 0.2) + 0.2 (1.0, 1.0, 0.5, 0.0) = \\ (0.8, 0.4, 0.0, 0.16) + (0.2, 0.2, 0.1, 0) = \mathbf{(1.0, 0.6, 0.1, 0.16)}$$

## Exercicis 8 i 9

Una forma d'expressar l'equació general del rendering és la següent:

$$L_o(\mathbf{x}, \omega_o, \lambda, t) = L_e(\mathbf{x}, \omega_o, \lambda, t) + \int_{\Omega} f_r(\mathbf{x}, \omega_i, \omega_o, \lambda, t) L_i(\mathbf{x}, \omega_i, \lambda, t) (\omega_i \cdot \mathbf{n}) d\omega_i$$

(a) Què creus que representa  $\lambda$  ?

Longitud d'ona de la llum

(b) Què representa  $\Omega$  ?



La semiesfera centrada al punt  $\mathbf{x}$  i alineada amb la normal  $\mathbf{n}$  que representa totes les possibles direccions en les que pot arribar llum a  $\mathbf{x}$ .

(c) Indica quin nom té la funció  $f_r$

És el BRDF.


(d) Què són els tres primers paràmetres de la funció  $f_r$ ?

Els paràmetres són: posició, direcció d'entrada, direcció de sortida.

### Exercici 10

Completa el següent FS per tal que calculi correctament el terme de Phong de la il·luminació:

```
uniform vec4 matAmbient, matDiffuse, matSpecular;
uniform vec4 lightAmbient, lightDiffuse, lightSpecular,
lightPosition; uniform float matShininess;

vec4 light(vec3 N, vec3 V, vec3 L)
{
    vec3 R = normalize( 2.0*dot(N,L)*N-
    L ); float NdotL = max( 0.0,
    dot( N,L ) ); float RdotV =
    max( 0.0, dot( R,V ) ); float Idiff
    = NdotL;
    float Ispec = 0;
    if (NdotL>0) Ispec = ........pow( RdotV, matShininess );

    return
        matAmbient * lightAmbient +
        matDiffuse * lightDiffuse * Idiff
        + matSpecular * lightSpecular *
        Ispec;
}
```

### Exercici 11

Indica, en la notació estudiada a classe,  $L(D|S)^*E$ , quins light paths són suportats

per:

(a) Raytracing clàssic

$LDS^*E$  i  $LS^*E$



(b) Two-pass raytracing

$LS^*DS^*E$  (i  $LS^*E$ )

### Exercicis 12 i 13

Volem generar amb RayTracing una imatge 256x256 d'una escena interior tancada. Els objectes de l'escena estan configurats de forma que la probabilitat de que qualsevol raig intersecti un mirall és de 0.5 (l'altre 0.5 correspon a un objecte difós).

a) Quants rajos primaris caldrà traçar?

$$256 \times 256 = 2^{16} \text{ rajos.}$$

b) Quants rajos reflectits caldrà traçar, en total, si admetem un únic nivell de recursivitat (per exemple LDSE)?

$$= 2^{15} = 32768$$



c) Quants rajos reflectits caldrà traçar, en total, si admetem dos nivells de recursivitat (per exemple LDSSE)?

$$2^{15} + 2^{14} = 49152$$

d) Quants rajos primaris caldrà traçar si volem antialiasing amb 4 mostres per píxel?

$$4 \times 256 \times 256 = 2^{18} \text{ rajos.}$$

### Exercici 14

Quin concepte de radiometria/fotometria és el més adient per mesura la quantitat d'energia per unitat de temps que arriba a una superfície, per unitat d'àrea (unitats W/m<sup>2</sup>)?

Irradiància



### Exercici 15

Indica, per cadascuna de les següents magnituds, si afecta (SI) o no (NO) a la direcció del raig transmès, d'acord amb la Llei de Snell (considera també efectes indirectes):

(a) Velocitat de propagació de la llum als medis SI

(b) Longitud d'ona de la llum SI



(c) Angle d'incidència SI

(d) Color difós de la superfície (Kd) NO

### Exercici 16

Tenim una aplicació que no suporta MipMapping, però volem simular el resultat de `GL_LINEAR_MIPMAP_LINEAR` en GLSL. Completa aquest codi, on `lambda` és un float amb el nivell de LOD (no necessàriament enter) més adient pel fragment:

```
vec4 sampleTexture(sampler2D sampler, vec2 texCoord, float lambda)
```

```
{  
    vec4 color0 = textureLod(sampler, texCoord, floor(lambda));  
    vec4 color1 = textureLod(sampler, texCoord, floor(lambda)  
+1);  
    return mix(color0, color1, fract(lambda));  
}
```

Podeu assumir que `textureLod(P,sampler,lod)` fa un accés bilineal a textura al punt `P` usant el nivell especificat a `lod`.

### Exercici 17

A classe hem estudiat un algorisme per simular reflexions especulars en miralls plans basat en objectes virtuals. Explica clarament per què és necessari, en general, fer servir la versió amb stencil buffer.



Per tal que els objectes virtuals (reflectits) apareguin només a la porció del pla ocupada pel mirall.

### Exercici 18

Amb la textura de l'esquerra, volem texturar l'objecte Plane com a la dreta.



Completa la línia que necessitem al VS:

```
vtexCoord = 4.0 * texCoord;
```

### Exercici 19

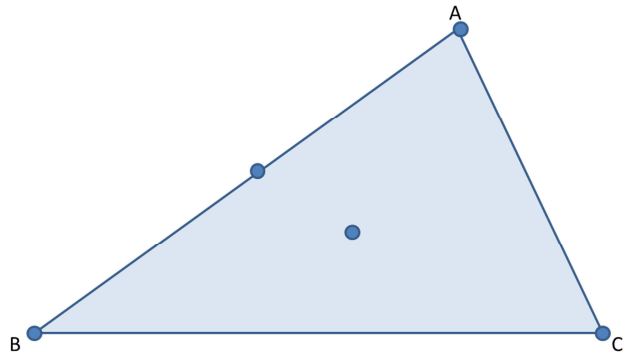
Indica les coordenades baricèntriques  $(\alpha, \beta, \gamma)$  associades als vèrtexs A, B, C del triangle, pels punts que

(a) Baricent

$(1/3, 1/3, 1/3)$

(b) Punt mig de l'aresta AB

$(0.5, 0.5, 0)$



(c) Si un punt P té coordenades  $(\alpha, \beta, \gamma)$  amb  $\alpha = 0.2$  i  $\beta = 0.3$ , què podem dir de  $\gamma$ ?

$\gamma = 0.5$  (han de sumar 1)

(d) Si un punt P té coordenades  $(\alpha, \beta, \gamma)$  amb  $\alpha < 0$ , què podem dir de P en relació al triangle?

P és exterior al triangle

### Exercici 20

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

---

Gràfics 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.

- Alpha Blending -

Geometry shader -

Rasterització

- Stencil Test

- Geometry shader

- Rasterització

- Stencil Test

- Alpha Blending

**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.

- Divisió de

perspectiva - Fragment

shader

- Pas a Clip Space

- Rasterització

Pas a clip space

Divisió de

perspectiva

Rasterització

Fragment shader

**Exercici 3**

El LOD 2 d'una textura té 256 x 128 texels. Quina mida té el LOD 0 d'aquesta mateixa textura?

**LOD 0 → 1024 x 512** (LOD 1 → 512 x 256; LOD 2 → 256 x 128)

#### Exercici 4

En quin espai de coordenades estan x, y en les crides GLSL dFdx, dFdy?

Window space.

#### Exercicis 5 i 6

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

M = modelMatrix	V <sup>-1</sup> = modelMatrixInverse
= viewingMatrix	P <sup>-1</sup> = viewingMatrixInverse
= projectionMatrix	N = projectionMatrixInverse
= normalMatrix	I = Identitat

- |                                                  |                                   |
|--------------------------------------------------|-----------------------------------|
| a) Pas d'un vèrtex de object space a model space | I                                 |
| b) Pas d'un vèrtex de object space a world space | M                                 |
| c) Pas d'un vèrtex de world space a eye space    | V                                 |
| d) Pas de la normal de object space a eye space  | N                                 |
| e) Pas d'un vèrtex de eye space a clip space     | P                                 |
| f) Pas d'un vèrtex de eye space a world space    | V <sup>-1</sup>                   |
| g) Pas d'un vèrtex de clip space a world space   | V <sup>-1</sup> * P <sup>-1</sup> |
| h) Pas d'un vèrtex de object space a clip space  | P * V * M                         |

### Exercici 7

Hem produït un fragment amb color RGBA = (1.0, 0.5, 0.0, 0.2) corresponent al pixel (i,j). El color RGBA del pixel (i,j) al buffer de color és (0.3, 0.1, 0.3, 0.7). Indica com hem de configurar la funció de blending si volem que el resultat sigui el color RGB (0.5, 0.2, 0.3). Justifica la resposta.

**glBlendFunc(...**

$$(1, 0.5, 0) \times \text{sfactor} + (0.3, 0.1, 0.3) \times \text{dfactor} = (0.5, 0.2, 0.3)$$

➔  $\text{dfactor} = 1$  ,  $\text{sfactor} = 0.2$

➔ **glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE)**

### Exercici 8

Escriu l'equació general del rendering, amb la formulació vista a classe, indicant clarament el tipus de radiància als diferents termes.

### Exercici 9

Indica, en la notació estudiada a classe,  $L(D|S)*E$ , quins light paths són suportats per:

(a) Raytracing clàssic

(b) Path tracing

### Exercici 10

Volem generar amb RayTracing una imatge 256x256 pixels d'una escena interior tancada. Quants shadow rays caldrà traçar, si tenim dos fonts de llum i els materials són difosos i opacs?



### Exercici 11

Completa el següent FS per tal que calculi correctament el terme especular (Phong) de la il·luminació:

```
uniform vec4 matDiffuse, matSpecular, lightDiffuse, lightSpecular;
uniform float matShininess;
```

```
vec4 light(vec3 N, vec3 V, vec3 L)
{
    vec3 R =
normalize( 2.0*dot(N,L)*N-L );
float NdotL = max( 0.0, dot( N,L ) );
float Idiff = NdotL;
float Ispec = 0;

    if{(NdotL>0)}
        float myDot = .....max( 0.0, dot( R,V ) );

        Ispec = .....pow( myDot, matShininess );
    }

return
    matDiffuse * lightDiffuse * Idiff
    + matSpecular * lightSpecular *
} Ispec;
```

### Exercicis 12 i 13

Completa aquest codi, corresponent als dos primers passos de l'algorisme de simulació d'ombres per projecció, amb stencil:

```
// 1. Dibuixa el receptor al color buffer i al stencil buffer
glEnable(GL_STENCIL_TEST);
glStencilFunc(GL_ALWAYS,1,1);
glStencilOp (GL_KEEP,GL_KEEP,GL_REPLACE);
dibuixa(receptor);

// 2.Dibuixa ocluser per netejar l'stencil a les zones a l'ombra
glDisable(GL_DEPTH_TEST);
glColorMask(GL_FALSE,...GL_FALSE);
glStencilFunc(GL_EQUAL, 1, 1);
glStencilOp (GL_KEEP, GL_KEEP, GL_ZERO);
glPushMatrix();
glMultMatrixf(MatriuProjeccio);
dibuixa(oclusor);
glPopMatrix();
```

#### Exercici 14

Indica com varia l'extensió de la penombra en els següents casos:

(a) La llum es puntual

**No hi ha penombra**

(b) Apropem ocluser i receptor de l'ombra

**Disminueix la penombra.**

#### Exercici 15

`glPolygonOffset(factor,units)` afegeix un offset a la z en window space d'acord a la següent fórmula:

$$Dz * factor + r * units$$

Com es calcula Dz i què representa?

$$Dz = \max(-\partial z / \partial x, \partial z / \partial y)$$

Representa quan tangencial és la primitiva a la direcció de visió.

#### Exercici 16

Tenim una aplicació que no suporta MipMapping, però volem simular el resultat de `GL_LINEAR_MIPMAP_LINEAR` en GLSL. Completa aquest codi, on `lambda` és un float amb el nivell de LOD (no necessàriament enter) més adient pel fragment:

```
vec4 sampleTexture(sampler2D sampler, vec2 texCoord, float lambda)
```

```
{
```

```
    vec4 color0 = textureLod(sampler, texCoord, floor(lambda));
```

```
    vec4 color1 = textureLod(sampler, texCoord, floor(lambda)+1);
```

```
    return
```

```
        mix(color0, color1, fract(lambda));
```

```
}
```

Podeu assumir que `textureLod(P,sampler,lod)` fa un accés bilineal a textura al punt P usant el nivell especificat a lod.

### Exercici 17

Què fa aquesta matriu?

$$\begin{bmatrix} 1-2a^2 & -2ba & -2ca & -2da \\ -2ba & 1-2b^2 & -2cb & -2db \\ -2ca & -2cb & 1-2c^2 & -2dc \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- (a) Projectió respecte una font posicional situada al punt (a,b,c)
- (b) Projectió respecte una font direccional situada al punt homogeni (a,b,c,d)
- (c) Reflexió respecte un pla (a,b,c,d)
- (d) Projectió ortogonal sobre el pla (a,b,c,d)

(c); n'hi ha prou amb provar amb el pla (0,1,0,0)

### Exercici 18

Indica en quin interval (el més ajustat possible) poden estar les coordenades Z dels punts interiors a la piràmide de visió d'una càmera perspectiva, segons l'espai considerat. Pots fer servir  $\pm\infty$  si s'escau.

- Object space:  $(-\infty, +\infty)$   
Eye space:  $(-\infty, 0)$   
NDC:  $[-1, 1]$   
Window space:  
 $[0, 1]$

Per alinear (establir la correspondència espacial) els models de volum obtinguts amb les diferents modalitats de captació d'imatge mèdica (MRI, CT...)

---

Gràfics 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.

- Fragment Shader
- Geometry shader
- Rasterització
- Stencil test

- Geometry shader
  - Rasterització
  - Fragment Shader
  - Stencil test

**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.

- dFdx, dFxy
- Divisió de  
perspectiva -  
Rasterització
- Vertex shader

- Vertex shader
  - Divisió de  
perspectiva -  
Rasterització
  - dFdx, dFxy

### Exercici 3

Sigui  $F(u,v)$  un *height field*. Si volem aplicar la tècnica de *bump mapping*, indica clarament què podem emmagatzemar per cada texel del bump map:

- (a) Si només disposem d'una textura amb un canal

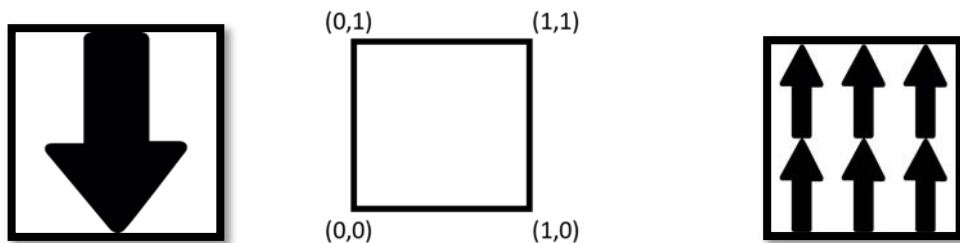
Directament  $F(u,v)$

- (b) Si disposem d'una textura amb dos canals

Gradient de  $F(u,v)$ :  $dF/du$ ,  $dF/dv$

### Exercici 4

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 = vec2(3,-2)*texCoord;  
  
    glPosition = vec4(vertex, 1.0);  
}
```

### Exercici 5

Tenim un FS que aplica una textura a l'objecte. Indica clarament quin efecte té incrementar el valor del uniform offset en la imatge resultant (suposa mode GL\_REPEAT):

```
uniform int offset = 0;  
  
...  
gl_FragColor = texture(sampler, vtxcoord + vec2(float(offset)))
```

Cap, ja que és un enter, i amb GL\_REPEAT només s'utilitza la part fraccionària de les coord de textura.

### Exercicis 6, 7, 8 i 9

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

- a) Pas de la normal de object space a eye space      N
- b) Pas d'un vèrtex de eye space a clip space      P
- c) Pas d'un vèrtex de eye space a world space       $V^{-1}$
- d) Pas d'un vèrtex de clip space a world space       $V^{-1} * P^{-1}$
- e) Pas d'un vèrtex de object space a clip space  
 $P * V * M$
- f) Pas d'un vèrtex de object space a model space      I
- g) Pas d'un vèrtex de object space a world space      M
- h) Pas d'un vèrtex de world space a eye space      V

### Exercici 10

Indica, en la notació estudiada a classe,  $L(D|S)*E$ , quins light paths són suportats per:

- (a) Raytracing clàssic

$LDS * E, LS * E$

- (b) Path tracing

$LS * DS * E$

### Exercicis 11 i 12

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

(a) Quin vector és paral·lel al raig primari

V

(b) Quin vector té la direcció del *shadow ray*?

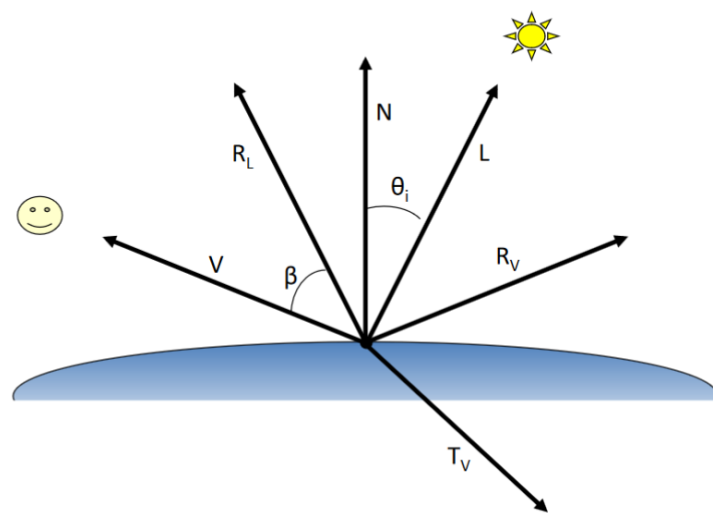
L

(c) Quin vector és paral·lel al raig reflectit?

$R_v$

(d) Què dos vectors determinen la contribució de

Phong?  $R_L$  i V

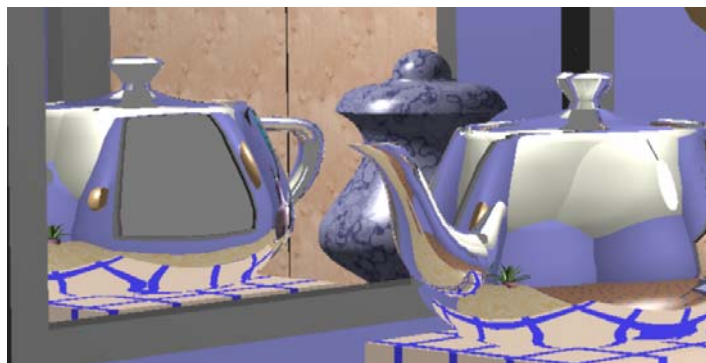


### Exercici 13

Escriu l'equació general del rendering, amb la formulació vista a classe, indicant clarament el tipus de radiància als diferents termes.

### Exercici 14

Considerant la figura:



(a) Amb quin algorisme s'ha generat? Ray Tracing

(b) Quin problema té clarament la imatge? Nivell de recursivitat massa baix -> manquen massa interreflexions.

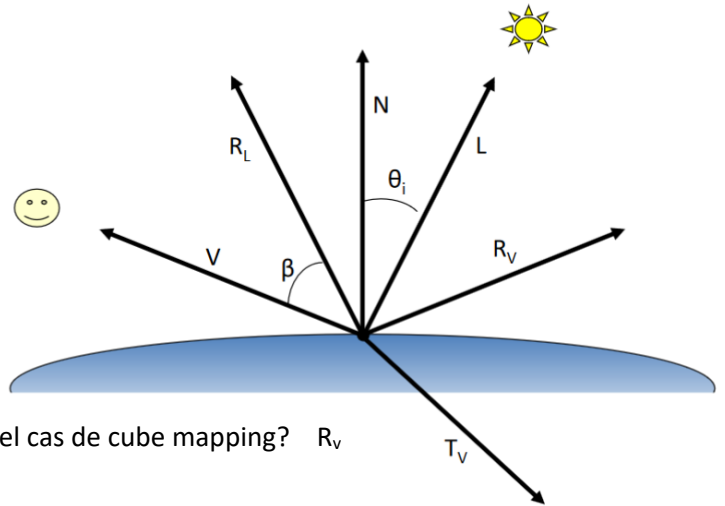
### Exercici 15

Quina unitat de radiometria, mesurada en  $\text{W}/\text{m}^2$  o en lux, es defineix com flux per unitat d'àrea?

Irradiància

### Exercici 16

Amb la notació de la figura:

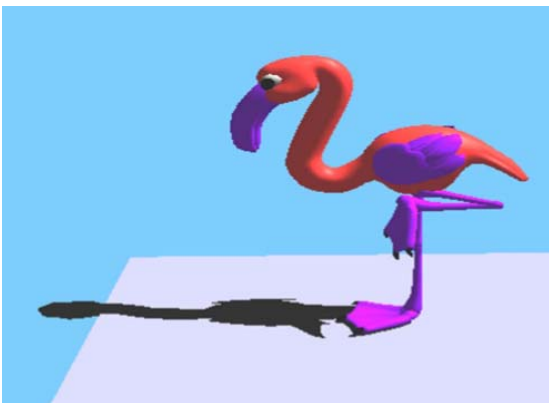


(a) Quin vector cal usar per indexar un cube map, en el cas de cube mapping?  $R_v$

(b) Quin dos vectors permeten calcular el terme de Lambert?  $N$  i  $L$

### Exercici 17

Indica com poder evitar aquest problema de la simulació d'ombres amb projecció:



Usant la versió amb stencil buffer per limitar el dibuix de l'ombra a la part ocupada pel receptor.

### Exercici 18

Explica en quines condicions la tècnica de mip mapping produeix una millora substancial de la qualitat de la imatge resultant.

Amb primitives texturades per les que cal un factor important de minification.



### Exercici 19

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

(c); n'hi ha prou amb provar amb el pla (0,1,0,0)

(a)

### Exercici 20

Indica quina és la diferència més important entre els models d'il·luminació local i els models d'il·luminació global.

Local → només llum directa

---

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

- Geometry shader

  - Rasterització
  - Fragment Shader
  - Depth test

**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

- Vertex shader
  - Clipping
  - Divisió de perspectiva
  - Rasterització

### Exercicis 3, 4, 5 i 6

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

M = modelMatrix	V <sup>-1</sup> = modelMatrixInverse
= viewingMatrix	P <sup>-1</sup> = viewingMatrixInverse
projectionMatrix	N = projectionMatrixInverse
N = normalMatrix	I = Identitat

- |                                                  |                   |
|--------------------------------------------------|-------------------|
| a) Pas d'un vèrtex de eye space a world space    | $V^{-1}$          |
| b) Pas d'un vèrtex de clip space a world space   | $V^{-1} * P^{-1}$ |
| c) Pas d'un vèrtex de object space a clip space  | $P * V * M$       |
| d) Pas d'un vèrtex de object space a model space | I                 |
| e) Pas d'un vèrtex de object space a world space | M                 |
| f) Pas d'un vèrtex de world space a eye space    | V                 |
| g) Pas de la normal de model space a eye space   | N                 |
| h) Pas d'un vèrtex de eye space a clip space     | P                 |

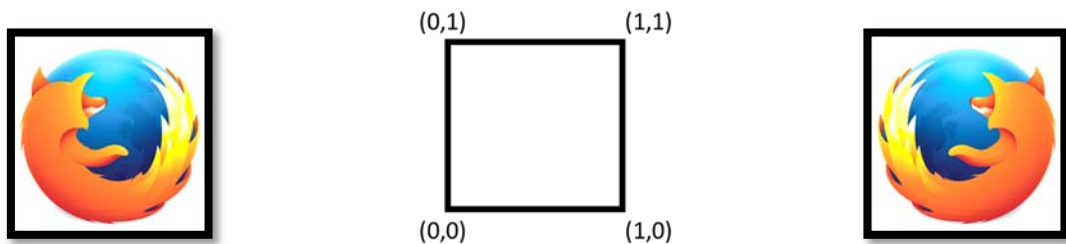
### Exercici 7

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        | NO |
| (b) Relief mapping       | SI |
| (c) Parallax mapping     | SI |
| (d) Displacement mapping | SI |

### 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() {  
  
    vtxCoord = vec2(-1,1)*texCoord;  
  
    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  $F(u,v)$ .

Bump mapping / Normal mapping

### 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      I
- (b) LDE           SI
- (c) LSE           SI
- (d) LDDSE       NO

### 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*?

L

- (b) Quin vector és paral·lel al raig reflectit?

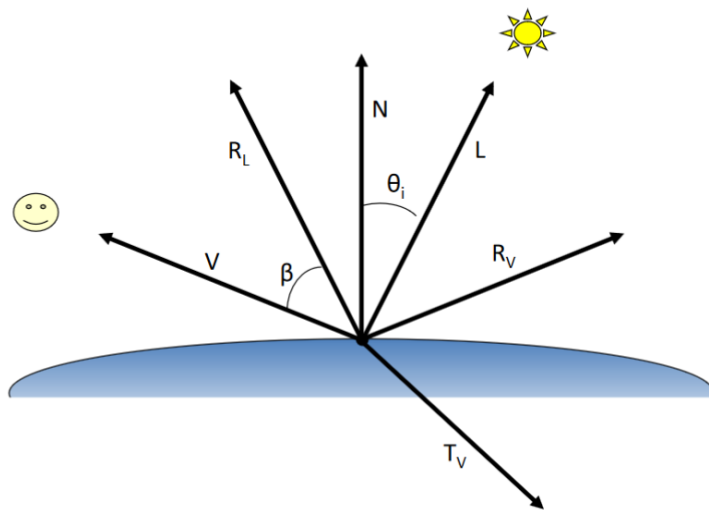
$R_v$

- (c) Què dos vectors determinen la contribució de

Lambert? L i N

- (d) Quin vector depèn de l'índex de refracció?

$T_v$



### 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  $\rho$ ? Una funció que decreix segons la distància

- (b) Com hauria de ser  $\rho$  per obtenir oclusió ambient? Funció constant  $\rho = 1$

### 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?

12 tri \* 3 v/tri = **36 vèrtexs** (hem de repetir vèrtexs perquè no comparteixen normals)

També és possible fer-ho només amb **24 vèrtexs** (cadascú dels 8 vèrtexs del cub només ha d'aparèixer amb 3 normals diferents; els dos triangles de cada cara poden re-usar un parell de vèrtexs).

### 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 = vtxCoord.st/vtexCoord.q;

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

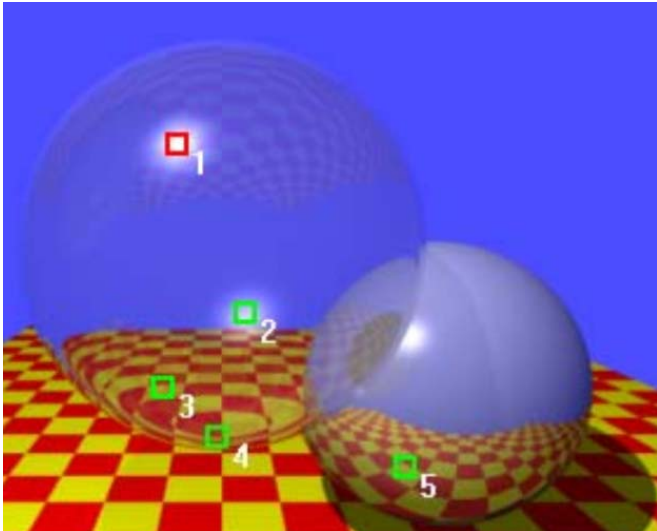
  float trueDepth = vtxCoord.p / vtxCoord.q;

  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"? LSE
- (b) Quin és el *light path* dominant que explica el color del píxel numerat amb un "5"? LSDE



### Exercici 18

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

- a) Shadow mapping, suposant que la llum és dinàmica

2 cops

- b) Reflexió amb objectes virtuals, amb stencil (ignoreu els passos en que només es dibuixa el mirall):

2 cops