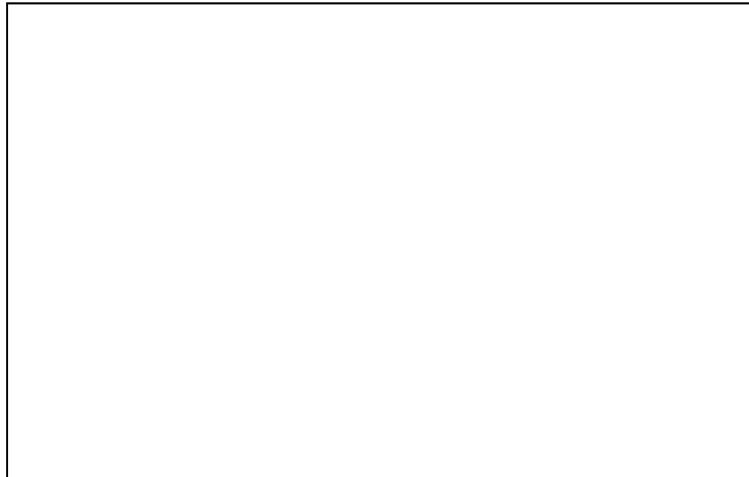


Nom:

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

**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

**Exercici 3**

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

Inicial:

Final:

Exercici 4

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

- (a) discard
- (b) EndPrimitive()
- (c) Escriure `gl_Position`
- (d) `dFdx`, `dFdy`

Exercici 5

Què coordenada del fragment modifica la funció `glPolygonOffset`?

En quin espai la modifica?

Exercici 6

Indica, amb la notació vista a classe, el *light path* que explica el color dominant del píxel indicat a la imatge.



Exercici 7

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

- (a) (0.2, -1.5, 1.8, 2) en clip space
- (b) (0, 0, 10, 1) en object space

Exercici 8

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

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

Exercici 9

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

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

Exercici 10

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

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

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

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);
```

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(sx,sy,sz) -> Scale matrix

T(tx,ty,tz) -> 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**.

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 Ω ?

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

Exercici 15

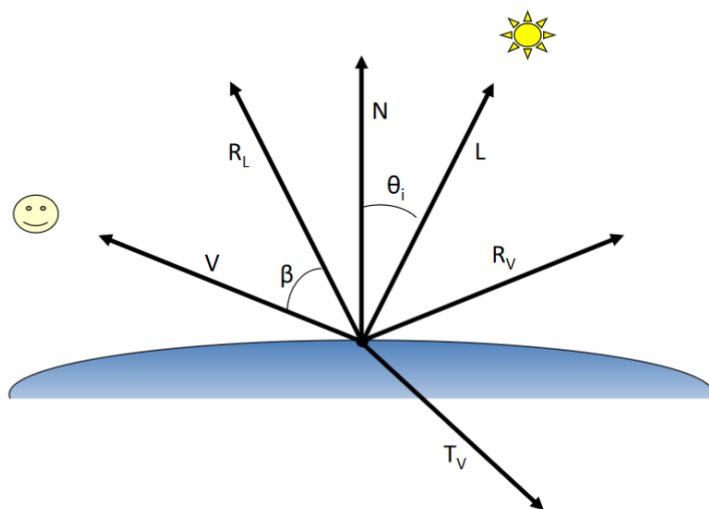
Indica quin tipus GLSL (float, vec2, vec3...) usaries per cada cas:

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

Exercici 16

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 transmès?



Exercici 17

Continuant amb la figura anterior...

- (a) Quin vector té la direcció del raig primari?
- (b) Quins dos vectors determinen la contribució local de Phong?

Exercici 18

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

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);
}
```