

Assigna a cada crida/tasca l'ordre relatiu (1,2,3,4) en que s'executa en un pipeline d'OpenGL sense GS:

[Cast]

Backface culling	4	✓
gl_Position is written	2	✓
VS execution starts	1	✓
Clipping to viewing frustum	3	✓

La resposta correcta és: Backface culling → 4, gl\_Position is written → 2, VS execution starts → 1, Clipping to viewing frustum → 3.

Assigna a cada crida/tasca l'ordre relatiu (1,2,3,4) en que s'executa en un pipeline d'OpenGL sense GS:

[Cast]

fragColor is written	3	✓
Rasterization	2	✓
Call to glDrawArrays	1	✓
Stencil Test	4	✓

La resposta correcta és: fragColor is written → 3, Rasterization → 2, Call to glDrawArrays → 1, Stencil Test → 4.

Assigna a cada crida/tasca l'ordre relatiu (1,2,3,4) en que s'executa en un pipeline d'OpenGL sense GS:

[Cast]

glGenVertexArrays	2	✗
Backface culling	1	✗
A new color is written to the color buffer	4	✓
glBufferData	3	✗

La resposta correcta és: glGenVertexArrays → 1, Backface culling → 3, A new color is written to the color buffer → 4, glBufferData → 2.

Assigna a cada crida/tasca l'ordre relatiu (1,2,3,4) en que s'executa en un pipeline d'OpenGL sense GS:  
[Cast]

Depth test

A new color is written to the color buffer

Clipping to viewing frustum

Call to glDrawArrays

La resposta correcta és: Depth test → 3, A new color is written to the color buffer → 4, Clipping to viewing frustum → 2, Call to glDrawArrays → 1.

#####  
#####

Quin terme o factor que apareix a l'equació general del rendering és el que retorna, de forma molt aproximada, un shader que implementa il·luminació de Phong?

[Cast]

Trieu-ne una:

- $Li(x,wi,t)$
- $wi \cdot n$
- $wi$
- $Lo(x,wo,t)$



La resposta correcta és:  $Lo(x,wo,t)$

**Lo(x,wo,t):** Aquesta és la **radància sortint** (outgoing radiance) en un punt x, en la direcció wo, en el temps t. És la quantitat de llum que surt d'una superfície cap a l'observador.

**Li(x,wi,t):** Aquesta seria la **radància entrant** (incoming radiance) que arriba al punt x des de la direcció wi.

**wi:** És la direcció de la llum incident (d'on ve la llum).

**wo:** És la direcció cap a l'observador (on va la llum que veiem).

Siguin

$P$  = punt visible de l'escena en una certa direcció de visió  $w$

$N$  = normal de la superfície en el punt  $P$

$L$  = vector unitari del punt  $P$  cap a la llum

En última instància, resoldre el problema de la il·luminació global, a nivell de cada punt de la imatge, equival a estimar...

[Cast]

Trieu-ne una:

- La irradiància de  $P$  en direcció  $w$
- $N \cdot L$
- La irradiància que arriba a  $P$
- La radiància de sortida al punt  $P$  en direcció  $-w$

La resposta correcta és: La radiància de sortida al punt  $P$  en direcció  $-w$

Siguin

$P$  = punt visible de l'escena en una certa direcció de visió  $w$

$N$  = normal de la superfície en el punt  $P$

$L$  = vector unitari del punt  $P$  cap a la llum

En última instància, resoldre el problema de la il·luminació global, a nivell de cada punt de la imatge, equival a estimar...

[Cast]

Trieu-ne una:

- $N \cdot L$
- $Lo(P, -w)$
- La intensitat emesa per  $P$
- La intensitat en  $P$  ✕

La resposta correcta és:  $Lo(P, -w)$

Siguin

$P$  = punt visible de l'escena en una certa direcció de visió  $w$

$N$  = normal de la superfície en el punt  $P$

$L$  = vector unitari del punt  $P$  cap a la llum

En última instància, resoldre el problema de la il·luminació global, a nivell de cada punt de la imatge, equival a estimar...

[Cast]

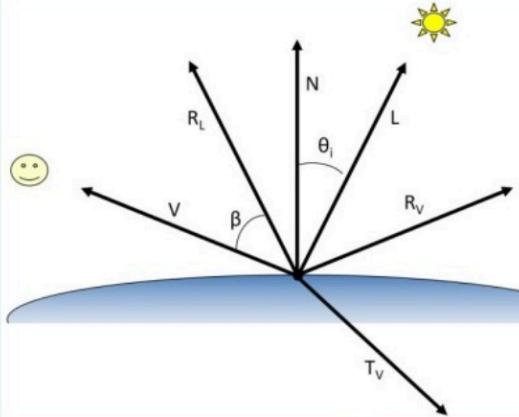
Trieu-ne una:

- La irradiància que arriba a  $P$
- La irradiància de  $P$  en direcció  $w$
- La radiància de sortida al punt  $P$  en direcció  $-w$
- $N \cdot L$

La resposta correcta és: La radiància de sortida al punt  $P$  en direcció  $-w$

#####  
#####

La figura mostra diferents vectors unitaris que intervenen en els càlculs d'il·luminació.



Quins vectors corresponen a direccions de rajos que traçaria two-pass raytracing, durant el *light pass*? (tria la més completa de les correctes)  
[Cast]

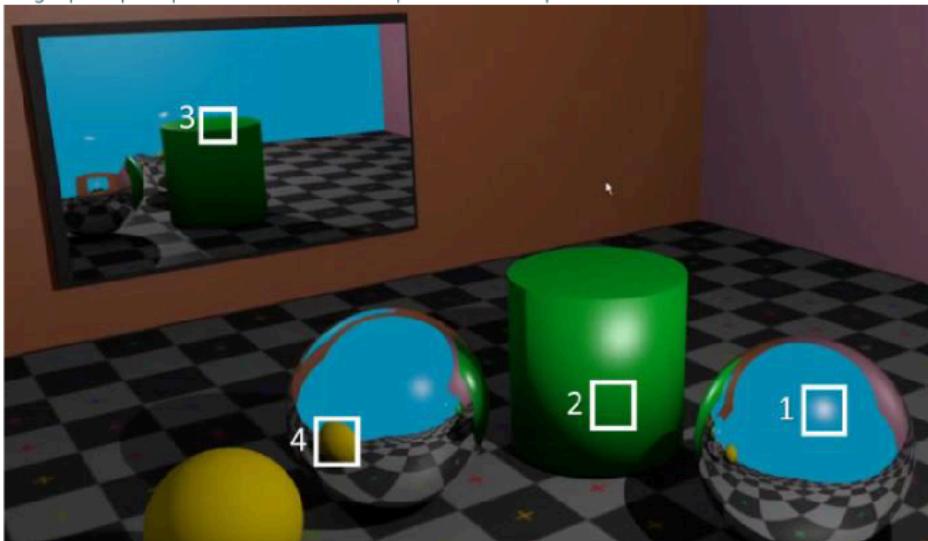
Trieu-ne una:

- $-L, R_V, T_V$
- $-L, R_L$
- $R_L, R_V$
- $-L, R_V$

La resposta correcta és:  $-L, R_L$

#####  
#####

El light path que explica el color dominant al pixel central del quadrat 3 és...



[Cast]

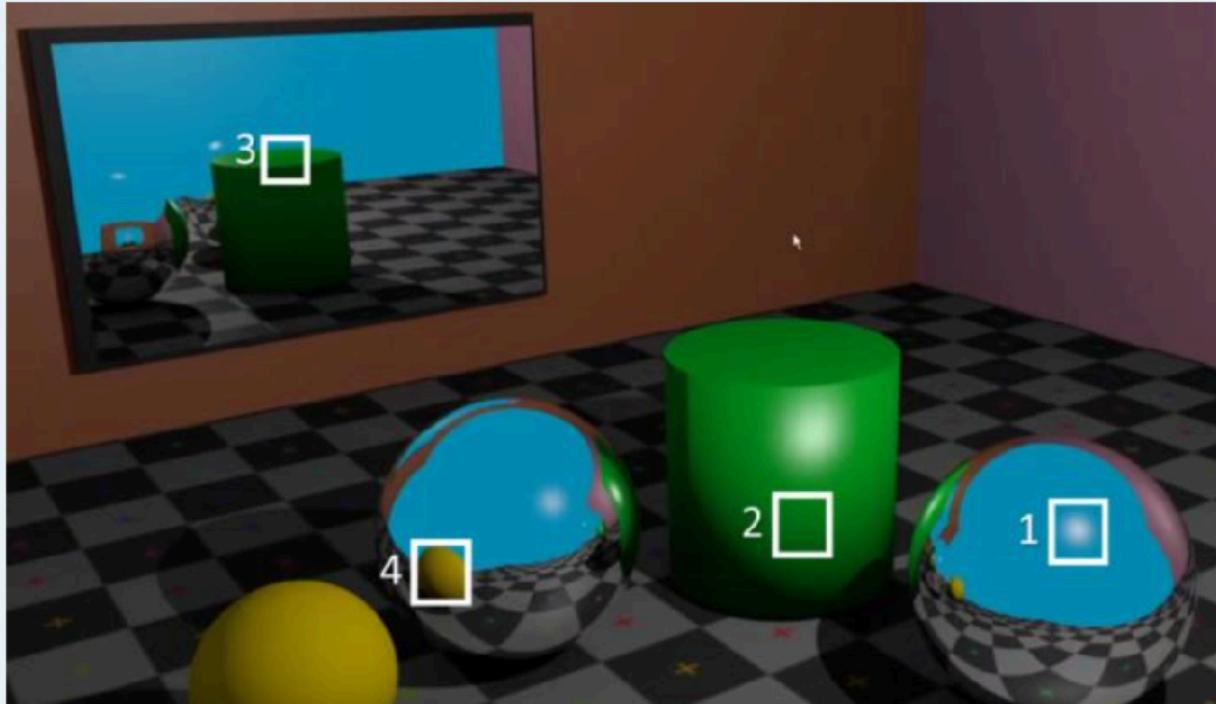
Trieu-ne una:

- LSDE
- LDSE
- LSDSE
- LSE



La resposta correcta és: LDSE

El light path que explica el color dominant al pixel central del quadrat **1** és...



[Cast]

Trieu-ne una:

- LSDSE
- LSDE
- LDDE
- LSE ✓

La resposta correcta és: LSE

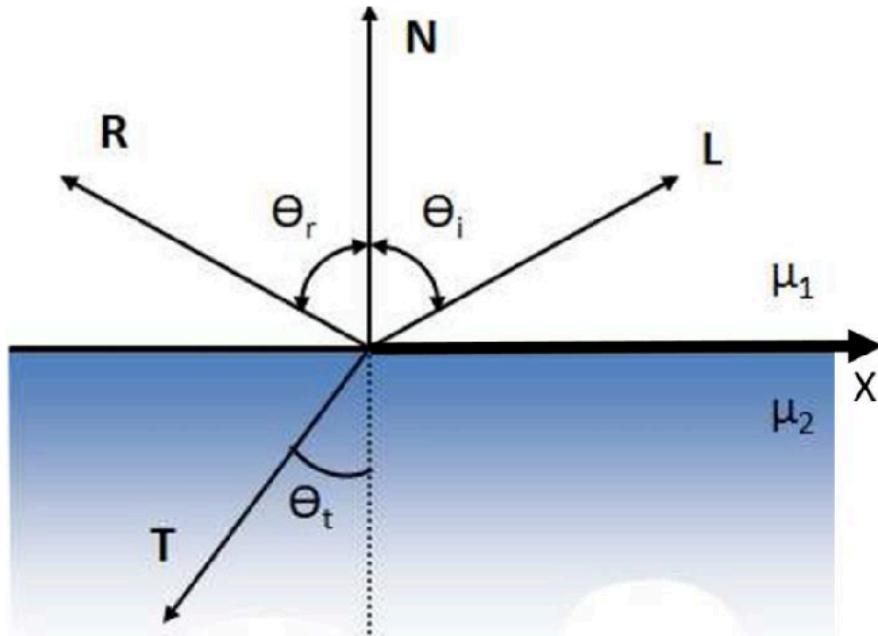
#####  
#####

Assuming que  $N = (0, 1, 0)$  i  $L = (0.44, 0.90, 0.00)$ , i que els dos medis tenen índex de refracció 1.30 i 2.50, calcula (amb el signe correcte) la component X del vector unitari trasmès T.

<https://atenea.upc.edu/mod/quiz/review.php?attempt=4921502&cmid=3802049>

19/3/23, 12:53

Examen final G (20 de gener 2023): Revisió de l'intent



[Cast]

Resposta: 0,2288



La resposta correcta és: -0,22666274506411

$$N = (0, 1, 0); L = (-0.44, 0.90, 0.00); \mu_1 = 1.30; \mu_2 = 2.50$$

$$\cos(\theta_i) = L \cdot N$$

$$\cos(\theta_i) = (-0.44)(0) + (0.90)(1) + (0.00)(0) = 0.90$$

$$\theta_i = \arccos(0.90) \approx 25.84^\circ$$

$$\mu_1 \sin(\theta_i) = \mu_2 \sin(\theta_r)$$

$$\sin(\theta_i) = \sin(25.84) \approx 0.4359$$

$$\sin(\theta_r) = (\mu_1/\mu_2) \times \sin(\theta_i) = (1.30/2.50) \times 0.4359 = 0.52 \times 0.4359 \approx 0.2267$$

$$\theta_r = \arcsin(0.2267) \approx 13.10$$

$$T = (\mu_1/\mu_2) \times L + [(\mu_1/\mu_2) \times \cos(\theta_i) - \cos(\theta_r)] \times N$$

$$T_x = (\mu_1/\mu_2) \times L_x + [(\mu_1/\mu_2) \times \cos(\theta_i) - \cos(\theta_r)] \times N_x$$

$$T_x = 1.30/2.50 * -0.44 + [1.30/2.50 \times \cos(25.84) - \cos(13.10)] * 0 = 0.52 * -0.44 + 0 = -0.2288$$

#####  
#####

Indica el valor de u (enter) que fa correcte aquest codi:

```
QImage img4("file.png");
QImage T5 = img4.convertToFormat(QImage::Format_ARGB32);
glGenTextures(1, &textureId6);
glBindTexture(GL_TEXTURE_2D, textureId6);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, T5.width(), T5.height(), 0, GL_RGBA, GL_UNSIGNED_BYTE, T5.bits());
g.glActiveTexture(GL_TEXTURE8);
g glBindTexture(GL_TEXTURE_2D, textureId6);
program->bind();
program->setUniformValue("textureMap", u); // sampler2D
```

[Cast]

Resposta:

8



La resposta correcta és: 8

Cal mirar quina textura s'activa amb la linea **g.glActiveTexture(GL\_TEXTURE8)**

#####  
#####

El punt 3D que resulta d'aplicar la transformació representada per la matriu  $\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 0 \end{bmatrix}$  al punt (18.00, 6.00, 18.00, 3.00) és...

<https://atenea.upc.edu/mod/quiz/review.php?attempt=4921502&cmid=3802049>

4/10

---

19/3/23, 12:53

Examen final G (20 de gener 2023): Revisió de l'intent

$$\begin{bmatrix} 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

[Cast]

Trieu-ne una:

- (6.00, 2.00, 18.00)
- (7.00, 5.00, 10.00) ✓
- (15.00, 21.00, 30.00)
- (21.00, 15.00, 30.00)

La resposta correcta és: (7.00, 5.00, 10.00)

Aquí cal multiplicar cada component xyz per el valor en la diagonal (18\*1, 6\*1, 18\*1, 3), dividir per la component w (18/3, 6/3, 18/3) i finalment sumar el desplaçament de l'última columna de la matriu a cada component (6+1, 2+3, 6+4) = (7,5,10)

El punt 3D que resulta d'aplicar la transformació representada per la matriu

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

al punt

(24.00, 8.00, 16.00, 4.00) és...

[Cast]

Trieu-ne una:

- (32.00, 24.00, 16.00)
- (16.00, 8.00, 24.00)
- (6.00, 8.00, 4.00)
- (24.00, 32.00, 16.00) ✗

La resposta correcta és: (6.00, 8.00, 4.00)

En una matriu amb la 4 columna neta no cal sumar desplaçaments

```
#####
#####
```

Amb les matrius indicades, calcula la distància al pla de retallat posterior (zfar):

$$\text{projectionMatrix} = \begin{bmatrix} 0.5 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.5 & 0.0 & 0.0 \\ 0.0 & 0.0 & -1.25 & 2.25 \\ 0.0 & 0.0 & -1.0 & 0.0 \end{bmatrix}$$
$$\text{projectionMatrixInv} = \begin{bmatrix} 2.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 2.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & -1.0 \\ 0.0 & 0.0 & 0.4444 & -0.5556 \end{bmatrix}$$

[Cast]

Resposta:  ✗

La resposta correcta és: 9

`projectionMatrixInv * [0, 0, 1, 1]T`

$x = 0$   
 $y = 0$   
 $z = -1$   
 $w = -0.1122$

$$-1/-0.1122 = 9$$

#####  
#####

Selecciona la única matriu de projecció (projectionMatrix) plausible per a una càmera perspectiva:  
[Cast]

Trieu-ne una:

- $\begin{bmatrix} 2.0 & 0 & 0 & 6.0 \\ 0 & 1.0 & 0 & 3.0 \\ 0 & 0 & 1.0 & 2.0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}$
- $\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.667 & 0.0 & 0.0 \\ 0.0 & 0.0 & -3.0 & -8.0 \\ 0.0 & 0.0 & -1.0 & 0.0 \end{bmatrix}$
- $\begin{bmatrix} 1.0 & 0 & 0 & 3.0 \\ 0 & 1.0 & 0 & 3.0 \\ 0 & 0 & 4.0 & 2.0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}$
- $\begin{bmatrix} 1.0 & 0 & 0 & 4.0 \\ 0 & 4.0 & 0 & 5.0 \\ 0 & 0 & 1.0 & 5.0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}$



La resposta correcta és:  $\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.667 & 0.0 & 0.0 \\ 0.0 & 0.0 & -3.0 & -8.0 \\ 0.0 & 0.0 & -1.0 & 0.0 \end{bmatrix}$

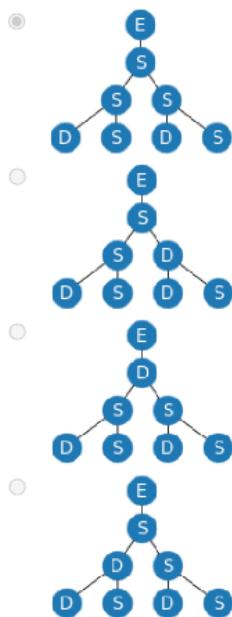
Una matriu de projecció sempre té components [2,2], [2,3], [3,2] negatives

#####  
#####

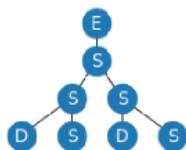
Indica quin arbre de rajos pot ser generat per Ray Tracing clàssic:

[Cast]

Trieu-ne una:



La resposta correcta és:



El ray tracing clàssic només es propaga per superfícies especulars (S), si un graf té una branca D que NO sigui fulla, aleshores no pot haver estat generat pel RayTracing clàssic

#####  
#####

Tenim una escena tancada que conté 49 objectes difosos i 3 llums puntuals. Volem generar una imatge de 640 x 1080 pixels amb Ray Tracing clàssic. Quants **shadow rays** cal llançar? (indica la resposta amb un enter)

[Cast]

Resposta:

La resposta correcta és: 2073600

els shadow rays es calculen de la següent forma

**Shadow rays = Nombre de pixels × Nombre de llums**

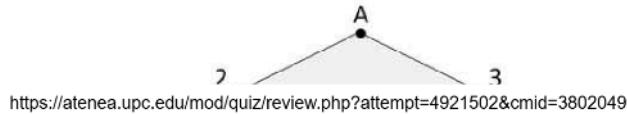
Shadow rays =  $691,200 \times 3 = 2,073,600$

**Primary rays = Nombre de pixels**

Primary rays =  $640 \times 1,080 = 691,200$  rays

#####  
#####

La figura mostra un triangle definit pels punts A, B, C.

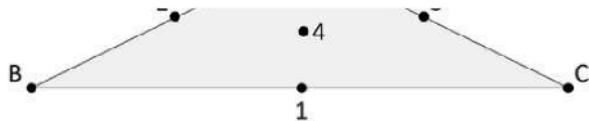


<https://atenea.upc.edu/mod/quiz/review.php?attempt=4921502&cmid=3802049>

5/10

19/3/23, 12:53

Examen final G (20 de gener 2023): Revisió de l'intent



Indica les coordenades baricèntriques (alfa, beta, gamma) del punt indicat amb un 4.

[Cast]

Trieu-ne una:

- (1/2, 0, 1/2)
- (1/2, 1/2, 1/2)
- (1/3, 1/3, 1/3)
- (1/2, 1/2, 0)

x

La resposta correcta és: (1/3, 1/3, 1/3)

La clau de l'exercici son les següents

1: La suma de coordenades baricèntriques ha de donar 1 és a dir  $\alpha + \beta + \gamma = 1$ .

2: P es pot expressar com una combinació lineal dels vèrtexs:  $P = \alpha \cdot A + \beta \cdot B + \gamma \cdot C$

Si el punt està exactament al vèrtex A, les seves coordenades baricèntriques seran  $(1, 0, 0)$ , perquè tot el "pes" està concentrat en A. Si el punt està al punt mitjà de l'aresta BC, les coordenades seran  $(0, 1/2, 1/2)$ , indicant que no hi ha contribució d'A però B i C contribueixen de manera igual.

El cas més interessant és el centroïde del triangle, que té coordenades baricèntriques  $(1/3, 1/3, 1/3)$ . Aquest punt està equidistant dels tres vèrtexs en termes de "influència" baricèntrica.

#####  
#####

Indica quina és l'opció més adient per a completar aquest codi a l'espai o espais indicats per '\_\_\_':

```
vec4 trace(Ray ray, float mu) {
    if ( intersectScene(ray, Phit, Nhit, muHit, Kd, Kr, Kt) )  {
        Ray shadowRay = Ray(Phit, normalize(lightPos-Phit));
        bool inShadow = intersectScene(shadowRay, aux);
        if (inShadow) shadow = 0.2; else shadow = 1.0;
        color+= shadow*light(Nhit, -ray.dir, lightPos-Phit, Kd, vec4(1.0));
        vec3 R = reflect(ray.dir, Nhit);
        color += Kr*trace(Ray(Phit, R), mu);
        vec3 T = refract(ray.dir, Nhit, _____);
        if (length(T)>0.0) color+=Kt*trace(Ray(Phit, T), muHit);
    }
    else color+=samplePanorama(ray.dir);
    return color;
}
```

[Cast]

Trieu-ne una:

- mu
- mu\*muHit
- mu/muHit
- muHit/mu

La resposta correcta és: mu/muHit

Aquesta divisió ens dona una proporció que indica "fins on hem viatjat al llarg del raig" dividit per "fins on hauríem de viatjar per arribar a la intersecció més propera".

Si aquesta proporció és menor que 1.0, significa que el punt actual del raig està més a prop de l'origen que la intersecció més propera, per tant aquest punt contribueix al color final. Si és major que 1.0, significa que estem més lluny que la intersecció més propera, i per tant aquest punt hauria d'estar ombrejat o no visible.

#####  
#####

Considera un sistema de cinema 360º basat en una pantalla gegant de forma esfèrica, de radi R=18, les quals reben imatges projectades per un projector 360º situat al centre. Si el projector emet 3000 lumens, distribuïts de manera uniforme en totes direccions, indica la il·luminància resultant en la pantalla.

[Cast]

Resposta:  ✗

La resposta correcta és: 0,73682844024026

$$\text{Il·luminància} = [\text{Lumens}/(4\pi)] / R^2$$

$$\text{Il·luminància} = 3000/(4\pi \times 18^2) = 0.73683$$

#####
#####

Indica quina és l'opció més adient per a completar aquest codi a l'espai o espais indicats per '\_\_\_':

```
// draw a scene containing opaque and semitransparent objects
void X::paintGL()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glEnable(GL_DEPTH_TEST);
    glDepthMask(GL_TRUE);
    opaque_objects.draw(); // unsorted
    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    _____;
    semitransparent_objects.draw(); // unsorted
    glDisable(GL_BLEND);
}
```

[Cast]

Trieu-ne una:

- glDisable(GL\_BLEND)
- glDepthMask(GL\_FALSE) ✓
- glColorMask(GL\_FALSE)
- glDepthMask(GL\_TRUE)

La resposta correcta és: glDepthMask(GL\_FALSE)

Quan dibuixes objectes transparents amb el depth mask activat (GL\_TRUE), cada fragment transparent que passa el depth test escriu la seva profunditat al buffer.

Quan poses glDepthMask(GL\_FALSE) abans de dibuixar objectes transparents, aquests fragments encara passen pel depth test normal, però no actualitzen el depth buffer. Això permet que múltiples capes transparents es puguin combinar correctament mitjançant l'alpha blending, ja que cada fragment transparent pot contribuir al color final sense "bloquejar" els fragments que vénen després.

#####  
#####

Per un determinat pixel  $(x,y)$ , els valors al frame buffer són:  $\text{depthBuffer}[x,y]=0.5$ ,  $\text{stencilBuffer}[x,y]=4$ . El test s'ha configurat amb `glStencilTest(GL_ALWAYS, 6, 255)`. Si es genera un fragment per aquest pixel, amb  $\text{gl_FragCoord.xyz} = (x, y, 0.6)$ , indica quin serà el resultat final al stencilBuffer, si l'operació està configurada amb: `glStencilOp(GL_ZERO, GL_INCR, GL_REPLACE)`;

[Cast]

Trieu-ne una:

- 6 ✕
- 5
- 4
- 0

La resposta correcta és: 5

$\text{depthBuffer}[x,y] = 0.5 \rightarrow$  valors mes grans fallen el Depth Test  
 $\text{stencilBuffer}[x,y] = 4 \rightarrow$  valor al que se li apliquen les Op  
`glStencilTest(GL_ALWAYS, 6, 255)` : `GL_ALWAYS`  $\rightarrow$  el stencil test sempre passa  
 $\text{gl_FragCoord.xyz} = (x, y, 0.6)$   
`glStencilOp(GL_ZERO, GL_INCR, GL_REPLACE)` x1  $\rightarrow$  falla stencil / x2  $\rightarrow$  passa stencil pero falla depth / x3  $\rightarrow$  passen Stencil i Depth

En aquest cas el Buffer, passa el test de Stencil (`GL_ALWAYS`), pero no el test de Depth (ja que  $0.6 > 0.5$ ), per tant s'executa la operació `GL_INCR`, que ens dóna un resultat de  $4+1 = 5$ .

`GL_REPLACE` canviaria el valor 5 per el 6 (2n paràmetre del `glStencilTest`)

#####  
#####

```

Donat aquest codi,
01 // 1. Draw mirror onto stencil buffer
02 glEnable(GL_STENCIL_TEST);
03 glStencilFunc(GL_ALWAYS, 1, 1);
04 glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE);
05 glDepthMask(GL_FALSE); glColorMask(GL_FALSE...);
06 draw(mirror);
07 // 2. Draw virtual objects
08 glDepthMask(GL_TRUE); glColorMask(GL_TRUE...);
09 glStencilFunc(GL_EQUAL, 1, 1);
10 glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);
11 ...
13 glCullFace(GL_FRONT);
14 draw(scene);
15 ...
16 // 3. Draw mirror
17 glDisable(GL_STENCIL_TEST);
18 ...
19 glCullFace(GL_BACK);
20 draw(mirror);
21 // 4. Draw real objects
22 draw(scene);

```

indica quina substitució de la línia 04 no alteraria la imatge resultant:

[Cast]

Trieu-ne una:

- glStencilOp(GL\_KEEP, GL\_INCR, GL\_KEEP);
- glStencilOp(GL\_ZERO, GL\_KEEP, GL\_REPLACE);
- glStencilOp(GL\_KEEP, GL\_REPLACE, GL\_KEEP);
- glStencilOp(GL\_KEEP, GL\_KEEP, GL\_ZERO);

La resposta correcta és: glStencilOp(GL\_ZERO, GL\_KEEP, GL\_REPLACE);

Mirem la linea 03 i 04

glStencilFunc(GL\_ALWAYS, 1, 1); -> stencil test SEMPRE PASSA

glStencilOp(GL\_KEEP, GL\_KEEP, GL\_REPLACE); -> x1 es irrelevante perque sempre passa

Per tant només s'ha de triar la opció amb els paràmetres de glStencilOp(, GL\_KEEP, GL\_REPLACE), o sigui, la opció b)

#####
#####  
#####

Donat aquest codi per generar reflexions en miralls,

```
// 1. Dibuixem el mirall a l'stencil buffer  
glEnable(GL_STENCIL_TEST);  
glStencilFunc(GL_ALWAYS, 1, mask);  
glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE);  
glDepthMask(GL_FALSE); glColorMask(GL_FALSE...);  
dibuixar(mirall);  
// 2. Dibuixem els objectes virtuals  
glDepthMask(GL_TRUE); glColorMask(GL_TRUE...);  
glStencilFunc(GL_EQUAL, 1, mask);  
glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);  
glPushMatrix(); glMultMatrix(matriu simetria)  
glLightfv(GL_LIGHT0, GL_POSITION, pos);  
glCullFace(GL_FRONT);  
dibuixar(escena);  
glPopMatrix();  
// 3. Dibuixem el mirall semitransparent  
glDisable(GL_STENCIL_TEST);  
glLightfv(GL_LIGHT0, GL_POSITION, pos);  
glCullFace(GL_BACK);  
dibuixar(mirall);  
// 4. Dibuixem els objectes reals  
dibuixar(escena);
```

un valor de la variable `mask` que preserva la correctesa del resultat és...

[Cast]

Trieu-ne una:

- 0
- 3 ✓
- 256
- 2

El valor no pot ser 0 ja que això implica no tenir màscara  
Tampoc pot ser 256 perquè la màscara és de 8 bits (255)  
A la línia 2 glStencilFunc(GL\_ALWAYS, 1, mask), implica que la mascara ha de validar el bit de menys pes (00000001), com que 2 en binari es 10, aquest bit de menys pes pasaria a estar inoperant, per tant només pot ser 3 (11).

```
#####
#####
```

En quin espai no és correcte interpolació simple (s,t) quan usen una càmera perspectiva?

[Cast]

Trieu-ne una:

- object space
- world space
- clip space
- NDC

La resposta correcta és: NDC

```
#####
#####
```

Siguin:

M: submatriu 3x3 de la modelMatrix

V: submatriu 3x3 de la viewMatrix,

P: submatriu 3x3 de la projectionMatrix,

la matriu que ens permet transformar **normals** de **object** space a **world** space es pot calcular com...

[Cast]

Trieu-ne una:

- $(MV)^{-1}$
- $M^{-T}$
- $M^T$
- $(MV)^{-T}$



La resposta correcta és:  $M^{-T}$

Siguin:

M: submatriu 3x3 de la modelMatrix

V: submatriu 3x3 de la viewMatrix,

la normalMatrix es pot calcular com...

[Cast]

Trieu-ne una:

- $(VM)^{-1}$
- $M^{-1}$
- $(VM)^{-T}$
- $(MV)^{-1}$

La resposta correcta és:  $(VM)^{-T}$

#####  
#####

L'expressió GLSL que representa l'expressió matemàtica  $K_d I_d (N \cdot L)$  és:

[Cast]

Trieu-ne una:

- matDiffuse \* lightDiffuse \* normalize(N) \* normalize(L)
- matDiffuse \* lightDiffuse \* dot(N,L)
- matDiffuse \* lightDiffuse \* cross(N,L)
- matDiffuse \* lightDiffuse \* N \* L

La resposta correcta és: matDiffuse \* lightDiffuse \* dot(N,L)

#####  
#####

Soposa que P és un punt en eye space. Una forma d'obtenir en un FS la direcció del vector normal en eye space és...

[Cast]

Trieu-ne una:

- normalize(cross(dFdx(P), dFdy(P)))
- dot(dFdx(P), dFdy(P))
- normalMatrix \* P
- normalize(dot(dFdx(P), dFdy(P)))

La resposta correcta és: normalize(cross(dFdx(P), dFdy(P)))

#####
#####

Indica la matriu que s'utilitza a la tècnica de shadow mapping per obtenir les coordenades de textura (s,t,p,q) d'un vèrtex, **si el vèrtex es troba en object space.**

[Cast]

Trieu-ne una:

- S(0.5)\*T(0.5)\*P
- T(0.5)\*S(0.5)\*P\*V\*M
- T(0.5)\*S(0.5)\*P\*V
- M\*P\*V



La resposta correcta és: T(0.5)\*S(0.5)\*P\*V\*M

L'ordre de les operacions es Mode IView Projection per passar de Obj.Space a ClipSpace.  
Després s'ha de escalar/rotar i per últim traslladar

Com que les operacions en OpenGL es fan al revés l'opcio correcta es la b)

Primer model (M), després view (V), seguit de projection (P). Finalment escalem (S) i traslladem (T)

T\*S\*P\*V\*M

#####
#####

Indica quina tasca/opció pot fer que alguns fragments no segueixin processant-se:  
[Cast]

Trieu-ne una:

- glClear
- glColorMask
- depth test ✓
- alpha Blending

La resposta correcta és: depth test

#####  
#####

Un estimador de la il·luminació global amb molta variància es distingeix per...  
[Cast]

Trieu-ne una:

- Genera imatges que requereixen noise filtering
- Suporta una gran varietat de camins de llum, especialment especulars.
- Té problemes per simular camins de llum especulars
- Suporta una gran varietat de materials (difosos i especulars)

La resposta correcta és: Genera imatges que requereixen noise filtering

Un estimador de la il·luminació global amb molta variància es distingeix per...

[Cast]

Trieu-ne una:

- Només simula camins perfectament especulars
- Suporta una gran varietat de camins de llum, especialment especulars.
- Genera imatges amb molt soroll
- Suporta una gran varietat de materials (difosos i especulars)

La resposta correcta és: Genera imatges amb molt soroll

#####  
#####

Indica el valor que ha de tenir la variable entera E en aquest fragment de codi, si estem dibuixant una malla de triangles amb 34 cares, 48 arestes i 15 vèrtexs:

```
g glBindVertexArray(VAO);
g glDrawElements(GL_TRIANGLES, E, GL_UNSIGNED_INT, (GLvoid*)0);
```

[Cast]

Resposta:



E es el nombre de vèrtexs, però té truc, s'agafa cada triangle (o cara) individualment, per tant alguns vèrtexs es repetiran, i els hem de tenir en compte tots, per tant **E = cares \* 3**  
 $E = 34 * 3 = 102$

#####  
#####

Indica el valor que ha de tenir la constant S en aquest fragment de codi:

```
const unsigned int S = ____;  
GLubyte read[S];  
glReadPixels(x, y, 2, 2, GL_RGBA, GL_UNSIGNED_BYTE, read);
```

[Cast]

Resposta: 16



La resposta correcta és: 16

S'ha de multiplicar els paràmetres p1, p2 i p3

p1 = 2

p2 = 2

p3 = RGBA, que té en compte les 4 components del color = 4

$$2 \times 2 \times 4 = 16$$

```
#####
#####
```

Diposem d'aquesta textura:



Indica amb quina opció el FS de sota obté aquest resultat amb l'objecte plane:



Recorda que plane.obj té coordenades de textura en [0,1].

```
fragColor = texture(colorMap, factor*vtexCoord + offset)
```

[Cast]

Trieu-ne una:

- factor=vec2(0.2, 0.2); offset=vec2(2.0, 0.1);
- factor=vec2(1.0, 0.2); offset=vec2(1.0, 1.0);
- factor=vec2(0.1, 1.0); offset=vec2(0.2, 0.0); ✓
- factor=vec2(0.0, 0.2); offset=vec2(1.0, 0.2);

La resposta correcta és: factor=vec2(0.1, 1.0); offset=vec2(0.2, 0.0);

Cal dividir la textura en 10 seccions, equivalent a multiplicar la component x per un factor de 0.1.

Com que volem la 3a textura, i per defecte  $x = 0.1$ , se li haurà de sumar 0.2, el que ens donarà  $x = 0.3$ , posició correcta.

```
#####
#####
```

Una superfície plana perfectament difosa rep una irradiància de  $8 \text{ W/m}^2$ . Si  $L$  és la radiància reflectida de sortida en direcció perpendicular a la superfície, i  $L'$  és la radiància reflectida de sortida a 70 graus de la normal, llavors...

[Cast]

Trieu-ne una:

- $L' = L / \cos(70)$  ✘
- $L = L'$
- $L' = L / \cos(20)$
- $L' = L * \cos(70)$

La resposta correcta és:  $L = L'$

#####  
#####

En aquest fragment de codi, la matriu M ha de ser...

```
// Pass 2. Draw the scene using the shadowmap
glClear(GL_DEPTH_BUFFER_BIT | GL_COLOR_BUFFER_BIT);
program->bind();
program->setUniformValue("lightViewMatrix", lightCamera.viewMatrix());
program->setUniformValue("lightProjectionMatrix", lightCamera.projectionMatrix());
QMatrix4x4 M = ____;
program->setUniformValue("biasMatrix", M);
program->setUniformValue("lightPos", lightCamera.getObs());
...
drawPlugin()->drawScene();
```

VS:

```
...
out vec4 vtexCoord;
void main()
{
    vtexCoord = biasMatrix*lightProjectionMatrix*lightViewMatrix*vec4(vertex,1.0);
    ...
}
```

[Cast]

Trieu-ne una:

- $$\begin{bmatrix} 0.5 & 0.0 & 0.0 & 0.5 \\ 0.0 & 0.5 & 0.0 & 0.5 \\ 0.0 & 0.0 & 0.5 & 0.5 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$
- $$\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.5 \\ 0.0 & 1.0 & 0.0 & 0.5 \\ 0.0 & 0.0 & 1.0 & 0.5 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$
- $$\begin{bmatrix} 0.5 & 0.0 & 0.0 & 0.25 \\ 0.0 & 0.5 & 0.0 & 0.25 \\ 0.0 & 0.0 & 0.5 & 0.25 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$
- $$\begin{bmatrix} 0.5 & 0.0 & 0.0 & 1.0 \\ 0.0 & 0.5 & 0.0 & 1.0 \\ 0.0 & 0.0 & 0.5 & 1.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

La resposta correcta és:

$$\begin{bmatrix} 0.5 & 0.0 & 0.0 & 0.5 \\ 0.0 & 0.5 & 0.0 & 0.5 \\ 0.0 & 0.0 & 0.5 & 0.5 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

La **biasMat** (matriu de bias) és la peça clau aquí. La seva funció és transformar coordenades del **clip space** (que va de -1 a 1) a **texture space** (que va de 0 a 1).

Pensem-ho així: quan la GPU calcula coordenades després de la projecció, aquestes estan en el rang [-1, 1] en cada eix. Però les textures s'accedeixen amb coordenades en el rang [0, 1]. Necessitem una transformació!

## La transformació matemàtica

Per convertir de [-1, 1] a [0, 1], necessitem:

- Escalar per 0.5 (per passar de rang 2 a rang 1)
- Després sumar 0.5 (per desplaçar de [-0.5, 0.5] a [0, 1])

Matemàticament: `texture_coord = (clip_coord × 0.5) + 0.5`

**Diagonal (0.5, 0.5, 0.5):** Realitza l'escalat

**Última columna (0.5, 0.5, 0.5, 1.0):** Realitza la translació

**Última fila (0, 0, 0, 1):** Preserva la coordenada homogènia

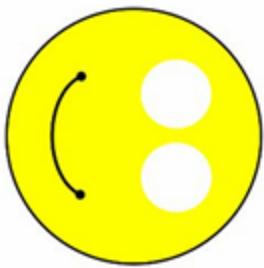
```
#####
#####
```

Indica el valor positiu més petit de la constant t per tal que el FS generi aquesta sortida amb l'objecte plane i la textura smile.

```
uniform sampler2D smile;
uniform float time;

const float PI = 3.141592653589793;
const float t = ???;

void main()
{
    vec2 p = vTexCoord;
    float a = 2 * PI * t;
    vec2 st = vec2( cos(a)*p.x - sin(a)*p.y, sin(a)*p.x + cos(a)*p.y);
    fragColor = texture(smile, st);
}
```



[Cast]

Resposta:  ✓

La resposta correcta és: 0,25

$$a = 2 * \pi * 0.25 = \pi/2 \text{ (90 graus)}$$

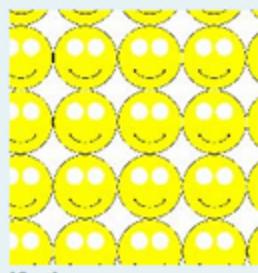
$$0.5 \rightarrow 180^\circ$$

$$0.75 \rightarrow 270^\circ$$

#####  
#####

Indica el valor de X per tal que el FS generi aquesta sortida amb l'objecte plane i la textura smile.

```
uniform sampler2D smile;  
  
const float X = ???;  
  
void main()  
{  
    vec2 p = vtexCoord;  
    vec2 st = p * X + vec2(1/X);  
    fragColor = texture(smile, st);  
}
```



[Cast]

Resposta:  ✓

La resposta correcta és: 4

#####  
#####

Les diferents etapes del pipeline d'OpenGL (VS, etc) comencen a executar-se quan s'invoca la funció...

[Cast]

Trieu-ne una:

- glBufferData()
- glDrawArrays()
- No vull contestar la pregunta
- glStart()
- glFinish()



La resposta correcta és: glDrawArrays()

Indica quina és l'aplicació de la tècnica de MipMapping:

[Cast]

Trieu-ne una:

- Speed-up frame buffer access
- Better texture sampling
- Zoom-in on textured objects
- Bicubic interpolation
- No vull contestar la pregunta

La resposta correcta és: Better texture sampling

Indica quina funció GLSL ens permet projectar els vèrtexs d'un objecte sobre una esfera unitària centrada a l'origen:

[Cast]

Gràfics 11 octubre 2022: Revisió de l'intent

<https://atenea.upc.edu/mod/quiz/review>

Trieu-ne una:

- project
- sphere
- reflect
- normalize
- No vull contestar la pregunta

La resposta correcta és: normalize

Indica en quina d'aquestes etapes del pipeline cal interpolar les sortides (variables **out**) del VS:

[Cast]

Trieu-ne una:

- Texture filtering
- Clipping
- Viewport transformation
- No vull contestar la pregunta
- Back face culling

La resposta correcta és: Clipping

Indica quina tasca/opció pot fer que alguns fragments no segueixin processant-se:

[Cast]

Trieu-ne una:

- glColorMask
- alpha test
- glClear
- glBind
- No vull contestar la pregunta

La resposta correcta és: alpha test

Indica el rang de valors de la coordenada z d'un punt visible en clip space:

[Cast]

Trieu-ne una:

- [-1,1]
- No vull contestar la pregunta
- [-z, z]
- [-w, w]
- [0, 1]

La resposta correcta és: [-w, w]

Indica quina expressió GLSL permet calcular el cosinus de l'angle incident (angle entre la normal i el light vector):

[Cast]

Trieu-ne una:

- cross(N,L)
- acos(N,L)
- dot(N, L)
- cos(N·L)
- No vull contestar la pregunta

La resposta correcta és: dot(N, L)

Indica el rang de valors de la coordenada z d'un punt visible en NDC space:

[Cast]

Trieu-ne una:

- [0, 1]
- [-w, w]
- [-z, z]
- No vull contestar la pregunta
- [-1,1]

La resposta correcta és: [-1,1]

Les coordenades de textura (s,t) que rep un VS en general seràn dins l'interval (tria l'opció correcta més restrictiva, en cas d'haver-ne)

[Cast]

Trieu-ne una:

- No vull contestar la pregunta
- (0,1)
- (-1, 1)
- $[-\infty, \infty]$
- [-1,1]

La resposta correcta és:  $[-\infty, \infty]$

#####  
#####

Donat el codi següent, indica el resultat de calcular una mostra de la textura al punt amb coordenades (s,t) = (3.5, 2.5).

```
// Dades de la textura (4 texels monocromàtics)
GLubyte textureData[4] = {
    40, // Texel (0,0) - lower left
    90, // Texel (1,0) - lower right
    180, // Texel (0,1) - upper left
    240 // Texel (1,1) - upper right
};

glEnable(GL_TEXTURE_2D);
GLuint textureID;
 glGenTextures(1, &textureID);
 glBindTexture(GL_TEXTURE_2D, textureID);
 glTexImage2D(
    GL_TEXTURE_2D,
    0, // mipmap level
    GL_LUMINANCE, // internal format
    2, // width
    2, // height
    0, // border
    GL_LUMINANCE, // buffer format
    GL_UNSIGNED_BYTE, // data type
    textureData // pointer to data
);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
```

[Cast]

Resposta:  ✗

```
color = (1-fx) * (1-fy) * texel(x,y) + (low_right)
      fx * (1-fy) * texel(x+1,y) + (low_left)
      (1-fx) * fy * texel(x,y+1) + (up_right)
      fx * fy * texel (x+1,y+1) + (up_left)
```

**Pas 1: Identificar les coordenades de mostreig** Les coordenades donades són (3.5, 2.5)

**Pas 2: Calcular les parts fraccionàries**

- $fx = 3.5 - 3 = 0.5$
- $fy = 2.5 - 2 = 0.5$

**Pas 3: Identificar els valors dels texels del codi** Observant l'array `textureData[ ]`:

- $textureData[0] = 40$  (texel 0,0 - lower left)

- `textureData[1] = 90` (texel 1,0 - lower right)
- `textureData[2] = 180` (texel 0,1 - upper left)
- `textureData[3] = 240` (texel 1,1 - upper right)

## Pas 5: Aplicar la fórmula bilineal

Unset

```
color = (1-0.5) * (1-0.5) * 90 +
       0.5 * (1-0.5) * 40 +
       (1-0.5) * 0.5 * 240 +
       0.5 * 0.5 * 180

color = 0.5 * 0.5 * 90 + 0.5 * 0.5 * 40 + 0.5 * 0.5 * 240 + 0.5 *
0.5 * 180

color = 0.25 * (90 + 40 + 240 + 180)

color = 0.25 * 550 = 137.5
```

#####
#####

Com a part d'un procés de generació de coordenades de textura, volem projectar un model centrat en el punt C sobre una esfera unitària. El mètode més adient és...  
[\[Cast\]](#)

Trieu-ne una:

- vec3 P = vertex.xyz - C; P = normalize(P); ✓
- vec3 P = (modelViewProjectionMatrix \* vertex).xyz;
- vec3 P = distance(vertex.xyz, C);
- vec3 P = vertex.xyz - C;

La resposta correcta és: `vec3 P = vertex.xyz - C; P = normalize(P);`

Indica en quina d'aquestes etapes del pipeline cal interpolar les sortides (variables **out**) del VS:  
[Cast]

Trieu-ne una:

- Perspective division
- Back face culling
- Viewport transformation
- Rasterization ✓

La resposta correcta és: Rasterization

Què fa aquest codi?

```
vec4 aux = modelViewProjectionMatrix * vec4(vertex, 1.0);  
vec3 foo = aux.xyz / aux.w;
```

[Cast]

Trieu-ne una:

- Projecta el vèrtex sobre una esfera
- Calcula la posició del vertex en model space
- Calcula la posició del vertex en eye space
- Passa vertex a clip space, i a continuació fa la divisió de perspectiva ✓

La resposta correcta és: Passa vertex a clip space, i a continuació fa la divisió de perspectiva

Un FS rep una variable **in vec3 P** amb la posició del fragment en *eye space*. Per calcular el *light vector* L cal usar...  
[Cast]

Trieu-ne una:

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

La resposta correcta és: `vec3 L = normalize(lightPosition.xyz - P);`

Indica com calcular la coordenada s per texturar una esfera unitària amb una projecció equirectangular, usant únicament el mapa de la dreta. Suposa que  $\theta = \text{atan}(\text{vertex}.x, \text{vertex}.z) + \pi$ , amb  $\theta \in [0, 2\pi]$ .



[Cast]

Trieu-ne una:

- `s = theta / (2\pi) ✗`
- `s = theta / (4\pi) + 0.5`
- `s = theta / (2\pi) + 0.5`
- `s = theta / (2\pi) * 2 + 2`

La resposta correcta és: `s = theta / (4\pi) + 0.5`

En el mipmap d'una textura de 128x128 texels, un texel del LOD 6 correspon a un grup de NxN texels del LOD 0. Indica el valor de N (enter):  
[Cast]

Resposta:  ✓

La resposta correcta és: 64

En una textura amb mipmaps, cada nivell (LOD - Level of Detail) té la meitat de resolució que l'anterior:

- LOD 0: 128×128 texels (textura original)
- LOD 1: 64×64 texels
- LOD 2: 32×32 texels
- LOD 3: 16×16 texels

- I així successivament...

## Resolent el problema pas a pas

**Primer**, necessitem entendre la relació matemàtica. Si el LOD 6 correspon a un grup de  $N \times N$  texels del LOD 0, això vol dir que cada texel del LOD 6 representa una àrea quadrada de  $N \times N$  texels de la textura original.

**Segon, calculem la resolució del LOD 6. Començant des del LOD 0 amb  $128 \times 128$ :**

- LOD 1:  $128 \div 2 = 64 \times 64$
  - LOD 2:  $64 \div 2 = 32 \times 32$
  - LOD 3:  $32 \div 2 = 16 \times 16$
  - LOD 4:  $16 \div 2 = 8 \times 8$
  - LOD 5:  $8 \div 2 = 4 \times 4$
  - LOD 6:  $4 \div 2 = 2 \times 2$

Per tant, el LOD 6 té una resolució de  $2 \times 2$  texels.

**Tercer**, determinem la relació de magnificació. Si la textura original és de  $128 \times 128$  i el LOD 6 és de  $2 \times 2$ , llavors:

- Cada texel del LOD 6 cobreix  $128 \div 2 = 64$  texels en cada dimensió de la textura original
  - Per tant, cada texel del LOD 6 representa un bloc de  $64 \times 64$  texels del LOD 0

#####

Indica el valor més gran que pot tenir la coordenada de textura s després del wrapping, si hem activat el mode GL\_CLAMP\_TO\_EDGE i la textura té una mida de 16x32 texels:  
[Cast]

Resposta: 1 

La resposta correcta és: 0,96875

En X: de 0.0 a 1.0 (correspondent als 16 texels)

En Y: de 0.0 a 1.0 (correspondent als 32 texels)

En una textura de 16 texels d'amplada, cada texel ocupa  $1/16 = 0.0625$  unitats de coordenada de textura.

El centre del darrer texel (el texel 15, comptant des del 0) està a la coordenada  $s = 15/16$ .

La resposta correcta és **0.96875** perquè representa la coordenada del centre del darrer texel abans que el clamping tingui efecte. Aquest valor es calcula com  $(16-0.5)/16 = 15.5/16 = 0.96875$ .

#####  
#####

En un determinat objecte, la coordenada s d'un fragment varia en (2, 5). Per convertir linealment aquesta coordenada per tal d'omplir l'interval (12, 20), la transformació correcta és...  
[Cast]

Trieu-ne una:

- $(s-2)/(5-2) * 20$
- $(s-2)/(5-2)*(20-12) + 12$  ✓
- $12 + (s-2)/(5-2) * (20-2)$
- $2 + (s-2)/(20-12) * 20$

La resposta correcta és:  $(s-2)/(5-2)*(20-12) + 12$

Indica la transformació geomètrica que **no** es pot aplicar com el producte d'una matriu **3x3** per un punt (x,y,z):

[Cast]

Trieu-ne una:

- rotació
- escalat no uniforme
- projecció ✓
- escalat uniforme

La resposta correcta és: projecció

#####  
#####

Indica el valor que retorna aquesta expressió GLSL:

`cross(vec3(3, 4, 2), vec3(2,2,4)).x`

[Cast]

Resposta:  ✗

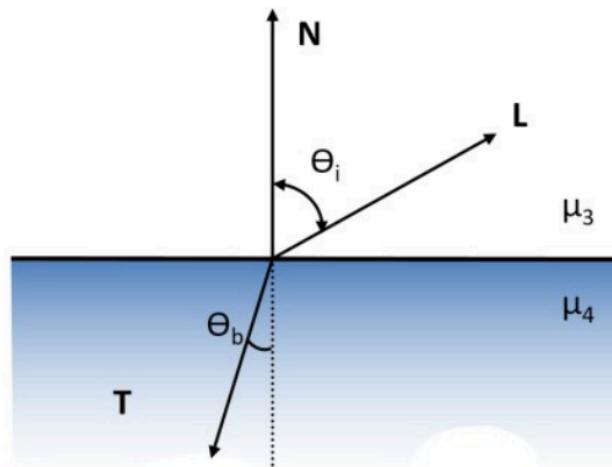
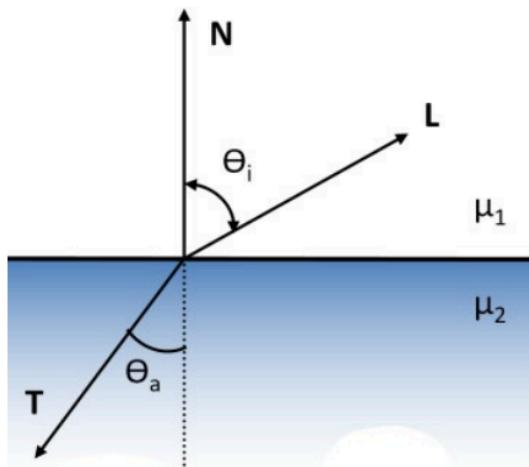
La resposta correcta és: 12

**Component X:**  $a_2b_3 - a_3b_2 = 4 \times 4 - 2 \times 2 = 16 - 4 = 12$

**Component Y:**  $a_3b_1 - a_1b_3 = 2 \times 2 - 3 \times 4 = 4 - 12 = -8$

**Component Z:**  $a_1b_2 - a_2b_1 = 3 \times 2 - 4 \times 2 = 6 - 8 = -2$

Considerant la figura



podem afirmar que... (tria la opció correcta més completa)  
[Cast]

Trieu-ne una:

- $(\mu_1 < \mu_2) \wedge (\mu_3 < \mu_4) \wedge (\mu_3/\mu_4 > \mu_1/\mu_2)$
- No vull contestar la pregunta
- $(\mu_1 < \mu_2) \wedge (\mu_3 < \mu_4) \wedge (\mu_2 > \mu_4)$
- $(\mu_1 < \mu_2) \wedge (\mu_3 < \mu_4) \wedge (\mu_1 > \mu_3)$
- $(\mu_2 > 0) \wedge (\mu_4 > 0) \wedge (\mu_1/\mu_2 > \mu_3/\mu_4)$

La resposta correcta és:  $(\mu_2 > 0) \wedge (\mu_4 > 0) \wedge (\mu_1/\mu_2 > \mu_3/\mu_4)$

A la parametrització equirectangular estudiada a classe, el punt amb coordenades esfèriques (en radians) (aproximadament) al punt de l'esfera...

[Cast]

Trieu-ne una:

- (0.13, 0.96, -0.24)
- (0.96, -0.24, 0.13)
- (1.24, 0.96, 0.13)
- No vull contestar la pregunta
- (-0.24, 0.96, 0.13)

La resposta correcta és: (-0.24, 0.96, 0.13)

Si volem crear una piràmide de mipmapping completa a partir d'una textura de 32x32 texels, quants nivell

[Cast]

Resposta:



La resposta correcta és: 6

El nombre total de nivells és  $\log_2(N) + 1$

#####
#####  
#####

Per a un determinat fragment, les derivades parcials de les coordenades de textura, un cop multiplicades p  
aquests valors:  $\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} = 64$ ;  $\frac{\partial u}{\partial y} = \frac{\partial v}{\partial x} = 0$ . Quin és el *LoD* més adient per a accedir a textura per a aq

## [Cast]

## Trieu-ne una:

- No vull contestar la pregunta
  - 12
  - 5

[atenea.upc.edu/mod/quiz/review.php?attempt=4362975&cmid=3499125](http://atenea.upc.edu/mod/quiz/review.php?attempt=4362975&cmid=3499125)

**si lees esto me debes un besito**

, 12:14

Qüestionari 21 març 2022: Attempt review

- 6  
3.0

La resposta correcta és: 6

El sistema de mipmapping utilitza aquestes derivades per determinar quin nivell de detall (LOD) és el més apropriat. La fórmula estàndard per calcular el LOD és:

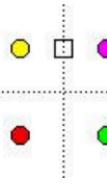
$$\text{LOD} = \log_2(\max(\sqrt{(\partial u / \partial x)^2 + (\partial u / \partial y)^2}, \sqrt{(\partial v / \partial x)^2 + (\partial v / \partial y)^2}))$$

Substituint els nostres valors:

- $\sqrt{(\partial u / \partial x)^2 + (\partial u / \partial y)^2} = \sqrt{(64^2 + 0^2)} = \sqrt{4096} = 64$
  - $\sqrt{(\partial v / \partial x)^2 + (\partial v / \partial y)^2} = \sqrt{(0^2 + 64^2)} = \sqrt{4096} = 64$

Per tant: LOD =  $\log_2(\max(64, 64)) = \log_2(64) = \log_2(2^6) = 6$

#####  
#####



La figura representa un grup de 2x2 texels, amb diferents colors RGB (interior de cada cercle): .....

retornarà aproximadament el color RGB...

[Cast]

Trieu-ne una:

- (1.00, 0.50, 0.50)
- (0.50, 0.50, 1.00)
- No vull contestar la pregunta
- (0.33, 0.50, 0.50)
- (1.00, 0.25, 0.50)

La resposta correcta és: (1.00, 0.50, 0.50)



La figura representa un grup de 2x2 texels, amb diferents colors RGB (interior de cada cercle): .....

..... Una mostra bilinial al quadrat

retornarà aproximadament el color RGB...

[Cast]

Trieu-ne una:

- (0.00, 1.00, 0.25)
- (0.00, 0.00, 0.50)
- (0.00, 0.50, 0.50)
- (0.00, 1.00, 0.50)
- No vull contestar la pregunta



La resposta correcta és: (0.00, 1.00, 0.50)

Cal fer la mitjana aritmètica dels colors entre els quals està el quadrat blanc

En la 1a imatge

$$\text{Groc} + \text{Magenta} / 2 = [(1,1,0) + (1,0,1)] / 2 = (1, 0.5, 0.5)$$

En la 2a imatge

$$\text{Verd} + \text{Cian} / 2 = [(0,1,0) + (0,1,1)] / 2 = (0, 1, 0.5)$$

#####  
#####

Disposem d'aquesta textura:



Volem texturar un polígon rectangular situat sobre el pla  $Z = 0$ . Sabem que el seu vèrtex mínim té coordenades  $(0,0,0)$ , i el vèrtex màxim té coordenades  $(1, 4, 0)$ . Si usem dos plans  $(S,T)$  per a generar les coordenades de textura, indica l'opció que permet texturar el polígon així (ignora la relació d'aspecte):



[Cast]

Trieu-ne una:

- S=vec4(3.00, 0.00, 0.00, 0.00); T=vec4(0.00, 1.00, 0.00, 0.00); ✓
- S=vec4(3.00, 0.00, 0.00, 0.00); T=vec4(1.00, 1.00, 1.00, 0.00);
- No vull contestar la pregunta
- S=vec4(0.33, 3.00, 0.25, 0.00); T=vec4(1.00, 3.00, 4.00, 0.00);
- S=vec4(1.00, 1.00, 0.00, 0.00); T=vec4(1.00, 0.25, 0.00, 0.00);

La resposta correcta és: S=vec4(3.00, 0.00, 0.00, 0.00); T=vec4(0.00, 1.00, 0.00, 0.00);

Disposem d'aquesta textura:



Volem texturar un polígon rectangular situat sobre el pla  $Z = 0$ . Sabem que el seu vèrtex mínim té coordenades  $(0,0,0)$ , i el vèrtex màxim té coordenades  $(2, 7, 0)$ . Si usem dos plans  $(S,T)$  per a generar les coordenades de textura, indica l'opció que permet texturar el polígon així (ignora la relació d'aspecte):



[Cast]

Trieu-ne una:

- S=vec4(2.00, 2.00, 2.00, 0.00); T=vec4(0.00, 0.50, 1.00, 0.00);
- S=vec4(2.00, 0.25, 0.50, 0.00); T=vec4(0.25, 0.25, 4.00, 0.00);
- S=vec4(2.00, 0.00, 0.00, 0.00); T=vec4(0.00, 0.29, 0.00, 0.00); ✓
- S=vec4(1.00, 0.29, 0.50, 0.00); T=vec4(1.00, 4.00, 4.00, 0.00);

La resposta correcta és: S=vec4(2.00, 0.00, 0.00, 0.00); T=vec4(0.00, 0.29, 0.00, 0.00);

### **Primer exemple:**

- Polígon: de (0,0) a (1,4) → Amplada = 1, Alçada = 4
- Veiem que es repeteix 3 vegades horitzontalment i 4 verticalment

### **Segon exemple:**

- Polígon: de (0,0) a (2,7) → Amplada = 2, Alçada = 7
- Veiem que es repeteix 2 vegades horitzontalment i aproximadament 2 verticalment

## **Pas 2: Calcular el factor d'escala**

La fórmula clau és: **Repeticions = Dimensió del polígon × Factor d'escala**

### **Per al primer exemple:**

- Horitzontalment: 3 repeticions en amplada 1 → Factor S = 3.00
- Verticalment: 4 repeticions en alçada 4 → Factor T = 1.00 (perquè  $4/4 = 1$ )

### **Per al segon exemple:**

- Horitzontalment: 2 repeticions en amplada 2 → Factor S = 1.00 (però veiem 2.00 a la resposta)
- Verticalment: ~2 repeticions en alçada 7 → Factor T ≈ 0.29 (perquè  $2/7 \approx 0.286$ )