Tria l'espai de coordenades en que ha d'estar P per tal que la transformació **projectionMatrixInverse*P** tingui

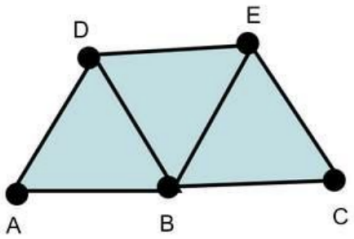
Select one:

- (●) clip space
- ( ) eye space
- ( ) object space
- ( ) world space

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

glGenVertexArrays	1
S'escriu gl_Position	2
Backface culling	3
Rasterització	4

Indica un ordre adient per emetre els vèrtexs dels triangles de la figura, en un GS, usant una única primitiva:



Select one:

- (●) ADBEC
- ( ) CEBAD
- ( ) DABEC
- ( ) ABDEC

Indica el punt que segur que serà FORA de la piràmide de visió d'una càmara perspectiva:

Select one:

- (●) (1.00, 5.00, 6.00, 2.00) en clip space
- ( ) (3.00, 4.00, 1.00, 11.00) en clip space
- ( ) (2.00, 0.00, -5.00, 1.00) en eye space
- ( ) (-2.00, -4.00, -8.00, 1.00) en eye space

Indica el valor que retorna aquesta expressió GLSL:

```
mix(9, 3, 0.7)
```

Answer:

Correct answers: [4.8]

Indica el valor que retorna aquesta expressió GLSL:

```
mod(7.9, 4)
```

Answer:

Correct answers: [3.9000000000000004]

Tenim una primitiva que ocupa tot un viewport de 2048x2048 pixels. Indica quin codi ens dona un cercle blanc de radi 295 pixels centrat al viewport:

Select one:

- fragColor = vec4(1-step(295, distance(gl_FragCoord.xy, vec2(1024)))
- fragColor = vec4(step(1024, distance(gl_FragCoord.xy, vec2(295)))
- fragColor = vec4(step(295, distance(gl_FragCoord.xy, vec2(2048)))
- fragColor = vec4(step(295, distance(gl_FragCoord.xy, vec2(1024)))

Soposa que
P és un punt,
N és la normal unitària en el punt,
(a,b,c,d) és el pla perpendicular a N que conté P,
L és un vector unitari cap a la font de llum,
R és el vector reflectit de L,
V és un vector unitari en direcció cap a la càmera, i
Q és un punt arbitrari.

Quina interpretació té l'expressió **cross(dFdx(P), dFdy(P))**?

Select one:

- Vector normal
- Projecció del vector posició sobre una esfera unitària
- Vector de reflexió especular de la llum directa
- Vector tangent a la superfície en el punt

Indica, en un FS, quina transformació de la z en window space té un efecte equivalent a invertir el depth test amb glDepthFunc(GL_GREATER):

Select one:

- gl_FragDepth = 1 - gl_FragCoord.z;
- gl_FragDepth = -1 * gl_FragCoord.z;
- gl_FragDepth = 0.5 * gl_FragCoord.z;
- gl_FragCoord.z = 1 - gl_FragCoord.z;

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)
```

Select one:

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

$$\begin{vmatrix} 1 & 2 & 3 \\ 6 & 5 & 4 \\ 7 & 8 & 9 \end{vmatrix} =$$

$$= 1 \cdot 5 \cdot 9$$
$$+6 \cdot 8 \cdot 3$$
$$+2 \cdot 4 \cdot 7$$
$$-3 \cdot 5 \cdot 7$$
$$-2 \cdot 6 \cdot 9$$
$$-4 \cdot 8 \cdot 1 =$$
$$= 45 + 144 + 56$$
$$-105 - 108 - 32 =$$
$$= 0$$

23.9 Escribe la ecuación del plano que pasa por los puntos: A(3,2,1), B(-4,-1,1) y C(-5,-3,-1):

Respuesta: $6x - 14y + 11z - 1 = 0$

Solución

Las coordenadas de un punto cualquiera del plano son: (x,y,z)

Las coordenadas del punto $A = (x_1, y_1, z_1)$ que corresponden a $(3,2,1)$.

Las coordenadas del punto $B = (x_2, y_2, z2)$ que corresponden a $(-4,-1,1)$.

Las coordenadas del punto $C = (x_3, y_3, z_3)$ que corresponden a $(-5,-3,-1)$.

Sustituyendo valores en:

$$\begin{vmatrix} x - x_1 & y - y_1 & z - z_1 \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \end{vmatrix} = 0$$

obtenemos:

$$\begin{vmatrix} x-3 & y-2 & z-1 \\ -7 & -3 & 0 \\ -8 & -5 & -2 \end{vmatrix} = 0$$

La matriu que representa una reflexió respecte un mirall triangular definit pels vèrtexs (6.00, 0.00, 9.00), (9.00, 0.00, 1.00), (7.00, 0.00, 1.00) és...

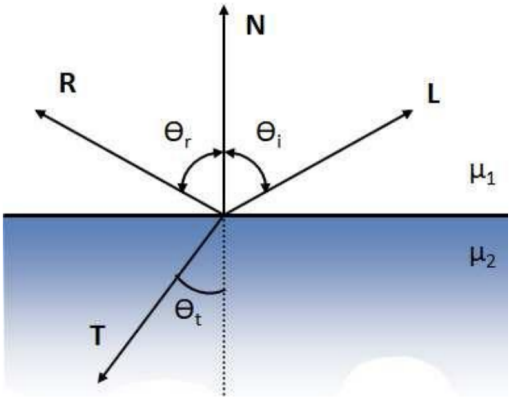
Select one:

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

Assigna a cada tasca l'ordre relatiu (1,2,3,4) en que s'executa, per simular reflexions especulars utilitzant la tècnica de sphere mapping en eye space:

El VS passa P, N a eye space	1
Es calcula el vector reflectit	2
Es calculen les coordenades (s,t) del fragment	3
Es pren una mostra de la textura que conté el sphere map	4

Tenint en compte la llei de Snell, indica quina parella de valors (μ1, μ2) explicarien de forma aproximada la direcció del raig trasmès que s'observa a la figura:



Select one:

- (1.09, 1.43)
- (1.43, 1.09)
- (0.09, 1.43)
- (1.09, 0.43)

Exercici 1

Copia a la dreta aquestes quatre les 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 les 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, dFxy	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 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(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**.

T(0.5)S(0.5)PV

Exercici 13

A l'equació general del rendering:

$$L_{\mathrm{o}}(\mathbf{x},\, \omega_{\mathrm{o}},\, \lambda,\, t) \;=\; L_{e}(\mathbf{x},\, \omega_{\mathrm{o}},\, \lambda,\, t) \;+\; \int_{\Omega} f_{r}(\mathbf{x},\, \omega_{\mathrm{i}},\, \omega_{\mathrm{o}},\, \lambda,\, t)\, L_{\mathrm{i}}(\mathbf{x},\, \omega_{\mathrm{i}},\, \lambda,\, t)\, (\omega_{\mathrm{i}} \cdot \mathbf{n})\, \mathrm{d}\,\omega_{\mathrm{i}}$$

Què representa Ω?

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

projectionMatrixInverse * **modelViewProjectionMatrix**

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 vtexCoord; // 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 = vtexCoord.st / vtexCoord.q;

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

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