Mini Apuntes G

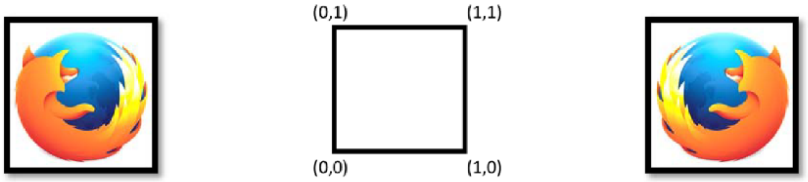
(cosas importantes)

1. Primero se hace el Vertex Shader, luego el Clipping, luego la división de perspectiva y por último la rasterización, seguidamente FS. (glDrawElements va antes que la escritura de gl_Position, luego div de perspectiva). El cálculo de iluminación se puede hacer tanto antes como después del rendering.

2. Primero va GS, luego Rasterización, luego FS y por último Depth Test (stencil test va antes del alpha blending)

3. El **heightfield** es qué tanto tenemos que desplazar la normal y esto se utiliza en relief mapping, parallax mapping y displacement mapping por ejemplo. No lo usamos en color mapping

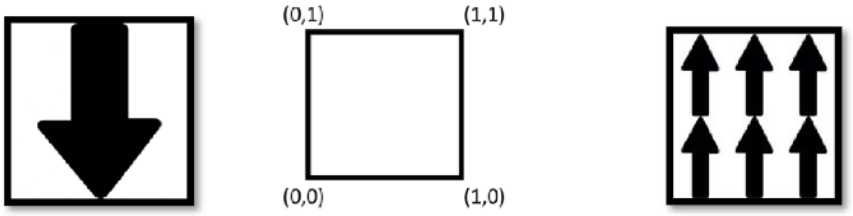
4. Amb la imatge de l'esquerra, volem texturar el quad del mig, per obtenir la imatge de la dreta:



Completa el següent VS per obtenir el resultat desitjat:

```
void main() {


      vtexCoord = vec2(-1,1)*texCoord;


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

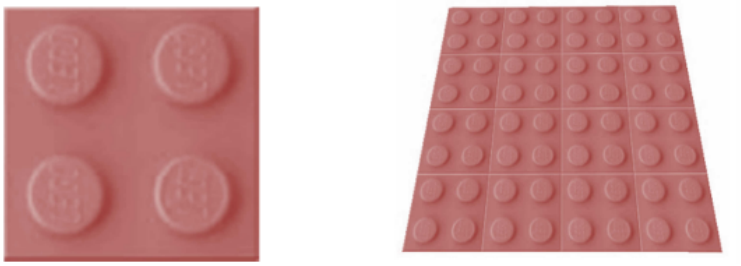
Amb la imatge de l'esquerra, volem texturar el quad del mig, per obtenir la imatge de la dreta:



Completa el següent VS per obtenir el resultat desitjat:

```
void main() {


vtexCoord = vec2(3,-2)*texCoord;


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

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;
```

5. **Bump Mapping** utiliza el gradiente de heightfield, **Normal Mapping** también

6. A parte de los que hay, LDE también es un camino soportado por two-pass raytracing OJO

(a) Raytracing clàssic

LDS*E i LS*E

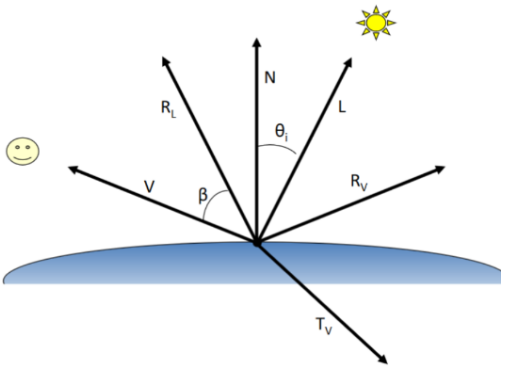
(b) Two-pass raytracing

LS*DS*E (i LS*E)

7.

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

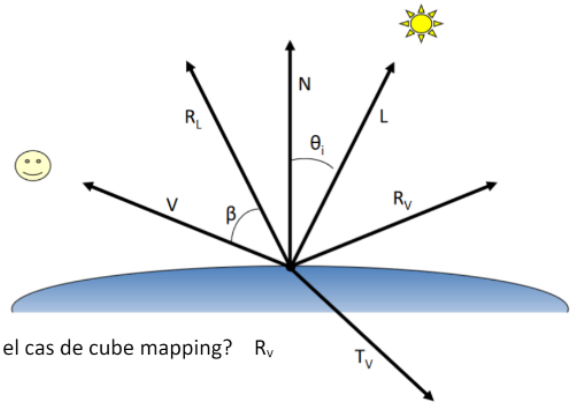
(c) Què dos vectors determinen la contribució de

Lambert?L i N

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

Tv

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

(d) Què dos vectors determinen la contribució de

Phong?R_L i V

8. Ecuación d'obscuràncies

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 ρ ? Una funció que decreix segons la distància

(b) Com hauria de ser ρ per obtenir oclusió ambient? Funció constant ρ = 1

9. Necesitamos 36 vértices para representar el VBO de un cubo, 12 triángulos * 3v/triángulo = 36v porque no comparten normales, aunque también lo podemos hacer con 24 vértices reutilizando vértices.

10. Código shadowmapping

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

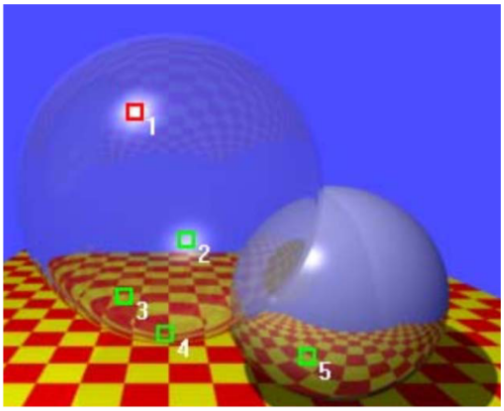
```
uniform sampler2D shadowMap;
uniform vec3 lightPos;
in vec3 N;
in vec3 P;
in vec4 vtexCoord; // coordenades de textura en espai
homogeni out vec4 fragColor;

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);
}
```

11. Light path dominante

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



12. La escena con shadow mapping y luces dinámicas se tiene que pintar 2 veces y la reflexión de objetos virtuales con stencil 2 también

13. Así se hacen los LOD:

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)

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

1+log2(512) = 10 (512x512, 256x256, 128x128, 64x64, 32x32, 16x16, 8x8, 4x4, 2x2, 1x1)

14. x,y en las llamadas de dFdx y dFdy están en window space.

15. Així se usa glBlendFunc

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) x sfactor + (0.3, 0.1, 0.3) * dfactor = (0.5, 0.2, 0.3)

➔ dfactor = 1 , sfactor = 0.2

➔ **glBlendFunc(GL_SRC_ALPHA, GL_ONE)**

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) = **(1.0, 0.6, 0.1, 0.16)**

16. Código de Phong:

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;
```

17. Código de Stencil:

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(
glStencilOp (
dibuixa(receptor);

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

18. glPolygonOffset:

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(∂z/∂x, ∂z/∂y)

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

19. Simulación GL_LINEAR_MIPMAP_LINEAR:

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.

20. Intervalos de coordenadas según el espacio:

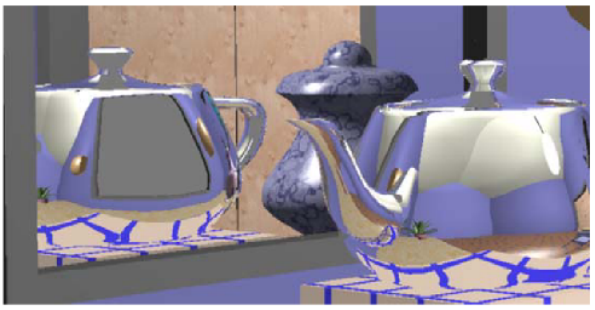
Indica en quin interval (el més ajustat possible) poden estar les coordenades Z dels punts interiors a la piràmide de visió d'una càmera perspectiva, segons l'espai considerat. Pots fer servir ±∞ si s'escau.

- Object space: (-∞,+∞)
  Eye space: (-∞,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...)

21. Error de la imagen, nivel de recursividad bajo

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.

22. Paths soportados

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

23. Lo que almacena cada texel en un bump map:

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

24. Cositas de GL_REPEAT e incrementar offsets:

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, vtexcoord + vec2(float(offset))
```

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

25. La ecuación del rendering:

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

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

(a) Què creus que representa λ ?

Longitud d'ona de la llum

(b) Què representa Ω ?

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

Completa l'eqüació general del rendering:

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

26. Cálculos de rayos de raytracing

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?

256x256 = 2^{16} 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 antialiàsing amb 4 mostres per píxel?

4x256x256 = 2^{18} rajos.

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? 1024x1024 = 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

27. Transformaciones de vistas

(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

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**:

28. Environment mapping:

Tenim aquest fragment de

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

(a) Quina tècnica està implementant?

Environment mapping

(b) En quin espai de coordenades està treballant?

Eye space; en cas contrari I = normalize(Pos-Obs)

29. Pintado en técnicas:

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

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