Examen Parcial de Gràfics Curs 2012-13 Q2

Nom i Cognoms:

Exercici 1 *(1 punt)*

Escriviu codi en llenguatge GLSL per:

a) Calcular un vector v en *object space* paral·lel a la direcció de visió de la càmera (és a dir, en la direcció de l'eix Z negatiu de la càmera).

```
vec3 v;
v =    ( gl_ModelViewMatrixInverse * vec4(0.0, 0.0, -1.0, 0.0)).xyz;
 o també
v = - gl_ModelViewMatrixInverse[2].xyz;
```

b) Passar el punt Pclip de *clip space* a *object space*

```
vec4 Pclip, Pobj;
...
Pobj = gl_ModelViewProjectionMatrixInverse * Pclip;
```

Exercici 2 *(1 punt)*

Escriviu codi en llenguatge GLSL per:

a) Passar una normal N de object space a eye space

```
vec3 N;
...
N = gl_NormalMatrix * N;
```

c) Passar un punt P de *eye space* a *model space*

```
vec4 P;
...
P = gl_ModelViewMatrixInverse * P;
```

Exercici 3 *(1 punt)*

Escriu codi GLSL per calcular les coordenades de la posició de la llum GL_LIGHT0 d'OpenGL en *world space* (assumint que en el moment d'execució del shader la transformació de modelat és la identitat).

```
vec4 P = gl_ModelViewMatrixInverse*gl_LightSource[0].position;
```

Exercici 4 *(1 punt)*

Suposeu que la matriu gl_ProjectionMatrix conté la matriu que genera la crida gluPerspective(fovy, aspect, n, f), que és:

$$\begin{bmatrix} \frac{\cot\frac{fovy}{2}}{aspect} & 0 & 0 & 0 \\ 0 & \cot\frac{fovy}{2} & 0 & 0 \\ 0 & 0 & \frac{n+f}{n-f} & \frac{2*n*f}{n-f} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Calcula el valor de les coordenades en clip space d'un vèrtex V que estigui situat exactament a la posició de l'observador (dóna el resultat final en funció de *n* i *f*).

Si està situat a la pos de l'observador, té coordenades (0,0,0,1) en eye space.

En clip space tindrà coordenades:

gl_ ProjectionMatrix * vec4(0,0,0,1) = darrera columna de la matriu anterior =

(0,0,2nf/(n-f), 0)

Té sentit fer la divisió de perspectiva per calcular les coordenades de V en normalized device space? Per què?

No, perquè (a) serà fóra de la piràmide de visió i (b) la component homogènia és 0.

Exercici 5 *(1 punt)*

Completa la següent funció per tal que calculi correctament la contribució especular usant el model de Phong, on N és la normal al punt, V és un vector del punt cap a la càmera, i L és un vector del punt cap a la font de llum (tots tres vectors en el mateix sistema de coordenades).

```
vec4 light(vec3 N, vec3 V, vec3 L)
{
  N=normalize(N); 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, gl_FrontMaterial.shininess );

  return
    gl_FrontMaterial.emission +
    gl_FrontMaterial.ambient  * gl_LightModel.ambient +
    gl_FrontMaterial.ambient  * gl_LightSource[0].ambient +
    gl_FrontMaterial.diffuse  * gl_LightSource[0].diffuse * Idiff+
    gl_FrontMaterial.specular * gl_LightSource[0].specular * Ispec;
}
```

Exercici 6 *(1 punt)*

Indica, per les següents accions, si són factibles (SI) o no (NO) dins un fragment shader:

(a) Modificar les coordenades x,y del fragment

NO

(b) Impedir que un fragment continuï en el pipeline

SI

(c) Calcular les coordenades del fragment en object space

SI

(d) Calcular la il·luminació utilitzant Blinn-Phong

SI

Exercici 7 *(1 punt)*

Completa el fragment shader de sota per tal que només es pintin els punts de l'objecte tals que la seva distància a la càmera sigui inferior a 3:

// Vertex shader

```
varying vec4 pos;
void main() {
  pos = gl_Vertex;
  gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
  gl_FrontColor = gl_Color;
}
```

// Fragment shader (a completar):

```
varying vec4 pos;
void main() {
  if (length((gl_ModelViewMatrix*pos).xyz)>3.0) discard;


  gl_FragColor = gl_Color;
}
```

Exercici 8 *(1 punt)*

Donat el següent fragment de codi:

```
uniform vec2 size; // mida de la textura, ex. (1024,1024)
...
vec2 uv = gl_TexCoord[0]*size;
vec2 dx = dFdx(uv);
vec2 dy = dFdy(uv);
float rho = max( dot(dx, dx), dot( dy, dy ) );
float lambda = max( 0.5 * log2(rho), 0.0 );
```

Explica clarament què fa aquest fragment de codi i per a què pot ser útil el valor lambda que calcula.

Estima la mida de la pre-imatge del fragment; lambda indica el nivell LOD de textura que cal utilitzar, quan fem servir mipmapping.

Exercici 9 *(1 punt)*

En relació a la tècnica de bump mapping, indiqui per cada afirmació si és certa o falsa:

(a) El fragment shader ha de descartar alguns fragments per tal de reproduir el detall del bump map a la silueta de l'objecte. FALS

(b) El fragment shader aplica els càlculs d'il·luminació amb la normal original pertorbada segons el bump map. CERT

(c) Tots els càlculs de bump mapping es fan en un vertex shader; no cal un fragment shader FALS

(d) La tècnica de bump mapping permet reproduir effectes de view-motion parallax FALS

Exercici 10 *(1 punt)*

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:

- Fragment Shader (FS)
- Rasterització
- Stencil test
- Vertex Shader (VS)

Vertex Shader (VS)

Rasterització

Fragment Shader (FS)

Stencil test