Examen Parcial de Gràfics Curs 2014-15 Q2

Nom i Cognoms:

Exercici 1

Aquest fragment de codi calcula el vector V que es necessita pels càlculs d'il·luminació:

```
vec3 V = normalize((gl_ModelViewMatrixInverse*vec4(0,0,0,1)).xyz – pos);
```

En quin espai(s) pot estar *pos* per què el càlcul sigui correcte?

En *object space (o world space si no hi ha transformació de modelat)*

Exercici 2

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:

- Crida a dFdx
- Depth test
- Escritura de gl_Position
- Rasterització

Escritura de gl_Position (VS)

Rasterització

Crida a dFdx (FS)

Depth test

(*) Amb early z-culling els dos últims canvien l'ordre

Exercici 3

Si N és un vector unitari en la direcció de la normal, i L és un vector unitari cap a la font de llum, quina interpretació geomètrica té aquest vector?

```
normalize( 2.0*dot(N,L)*N-L );
```

És un vector unitari en la direcció de la reflexió de L respecte N.

Exercici 4

Indica i declara en GLSL els dos vectors cal que el VS passi al FS per tal d'usar el model de Phong al FS.

```
varying vec3 normal;
```

```
varying vec3 pos;
```
(vec4 també és correcte)

Exercici 5

Completa la segona línia d'aquest fragment de codi que avalua la part especular de la il·luminació:

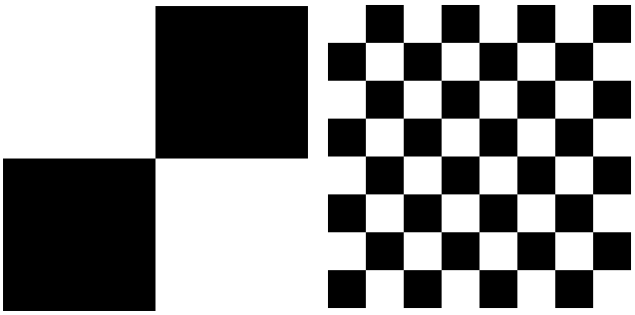
```
float RdotV = max( 0.0, dot( R,V ) );

float Ispec = pow( RdotV,  .....................................................);
```

gl_FrontMaterial.shininess

Exercici 6

Completa aquest FS de forma que, amb la textura de l'esquerra, produeixi un tauler d'escacs com el de la dreta, quan s'aplica a l'objecte plane, que té coordenades de textura del (0,0) al (1,1).



```
void main() {
        vec2 f = vec2(....................., ......................);
        gl_FragColor = texture2D(c, f * gl_TexCoord[0].st);
}
```

vec2(4, 4)

Exercici 7

Aquest fragment de codi OpenGL està fent servir la generació automàtica de coordenades de textura amb el mode GL_OBJECT_LINEAR:

```
GLint s_plane[4] = { 5, 0, 0, 0 };
GLint t_plane[4] = { 0, 2, 0, 0 };
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR);
glTexGeniv(GL_S, GL_OBJECT_PLANE, s_plane);
glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR);
glTexGeniv(GL_T, GL_OBJECT_PLANE, t_plane);
glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);
```

Quines coordenades de textura (s,t) es calcularan pel vèrtex (10, 2, 127, 1)?

-> (5*10, 2*2) = (50, 4)

Exercici 8

Què fa aquest codi? En quina tècnica ens pot ser útil?

```
float F =  texture2D(height,gl_TexCoord[0].st+vec2(0.0,0.0)).r;
float Fx = texture2D(height,gl_TexCoord[0].st+vec2(eps,0.0)).r;
float Fy = texture2D(height,gl_TexCoord[0].st+vec2(0.0,eps)).r;
vec2 dF = 0.05*vec2(Fx-F, Fy-F)/eps;
```

Calcula una aproximació del gradient d'un heightfield codificat en una textura. És útil en bump mapping.

Exercici 9

Què creus que fa aquest fragment de codi?

```
float foo() {
   vec2 uv = vec2(WIDTH, HEIGHT) * gl_TexCoord[0].st;
   vec2 dx = dFdx(uv);
   vec2 dy = dFdy(uv);
   float rho = max( dot(dx, dx), dot( dy, dy ) );
   return max( 0.5 * log2(rho), 0.0 ); }
```

Retorna el paràmetre lambda que indica el nivell de LOD requerit per MIPMAPPING.

Exercici 10

Què és la pre-imatge d'un fragment?

És la regió de la textura associada a la regió rectangular associada al fragment, d'acord amb el mapping definit per les coordenades de textura de la primitiva.

Exercici 11

La matriu necessària per implementar projective texture mapping és pot obtenir amb aquest codi:

```
glLoadIdentity();
glTranslated(0.5, 0.5,0.5);
glScaled(0.5, 0.5, 0.5);
gluPerspective(…);
gluLookAt(…);
```

Completa aquest fragment de codi per obtenir exactament la mateixa matriu:

```
glLoadIdentity();
glScaled(0.5, 0.5, 0.5);

glTranslated(....., ....., .....);                    glTranslated(1 , 1 , 1);

gluPerspective(…);
gluLookAt(…);
```

Exercici 12

Si (T, B, N) són els vectors tangent, bitangent i normal, expressats en **object space**, indica quina és la matriu que ens permet passar de **tangent space** a **object space**, considerant aquesta definició:

```
mat4 M = mat4(T, B, N, vec4(0,0,0,1));
```

Directament la matriu M

Exercici 13

Si implementem la tècnica de bump mapping fent servir un normal map, en quin espai estaran codificades les normals al normal map?

En tangent space

Exercici 14

Amb la textura de l'esquerra hem texturat l'objecte Plane que veiem al viewport com a la imatge de la dreta.

Indica clarament quines derivades parcials de gl_TexCoord[0].s i gl_TexCoord[0].t seran 0.

dFdx(s) i dFdy(t)

Exercici 15

Escriviu codi en llenguatge GLSL per passar un punt P de *clip space* a *model space*

```
vec4 P;
...
P = gl_ModelViewProjectionMatrixInverse * P;
```

Exercici 16

Quin problema veus en aquest fragment de codi GLSL?

```
vec3 V = normalize(-gl_Position).xyz;
```

V no és unitari per què s'ha normalitzat un vec4; hauria de ser vec3 V = normalize(-gl_Position.xyz);