Examen Final de Gràfics Curs 2011-12 Q2

Nom i Cognoms:

Exercici 1 *(1 punt)*

Completa el següent codi GLSL (vigila la sintaxi!):

(a) Funció que donat un vèrtex (en *eye space*) i una llum posicional, retorna el vector unitari L (també en *eye space*) que cal per calcular la il·luminació:

```
vec3 lightVector(vec3 vertex, gl_LightSourceParameters light)
{
      return normalize( light.position.xyz - vertex );
}
```

(b) Il·luminació bàsica a nivell de fragment:

```
// Vertex shader
varying vec3 normal;

void main() {
      normal = gl_NormalMatrix * gl_Normal;
      gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
      gl_FrontColor  = gl_Color;
}

// Fragment shader
varying vec3 normal;

void main() {
      gl_FragColor = normal.z * gl_Color;
}
```

(c) Completa aquest shader per tal que calculi el color del fragment com el color del texel que indiquen les seves coordenades de textura (texture unit 0):

```
uniform sampler2D sampler;

void main(void) {
  gl_FragColor = texture2D( sampler, gl_TexCoord[0].st);
}
```

Exercici 2 *(1 punt)*

Completa el geometry shader d'aquest programa perquè, a més a més de l'objecte 3D, dibuixi la seva reflexió respecte el pla horitzontal Y=0 (veure figura).

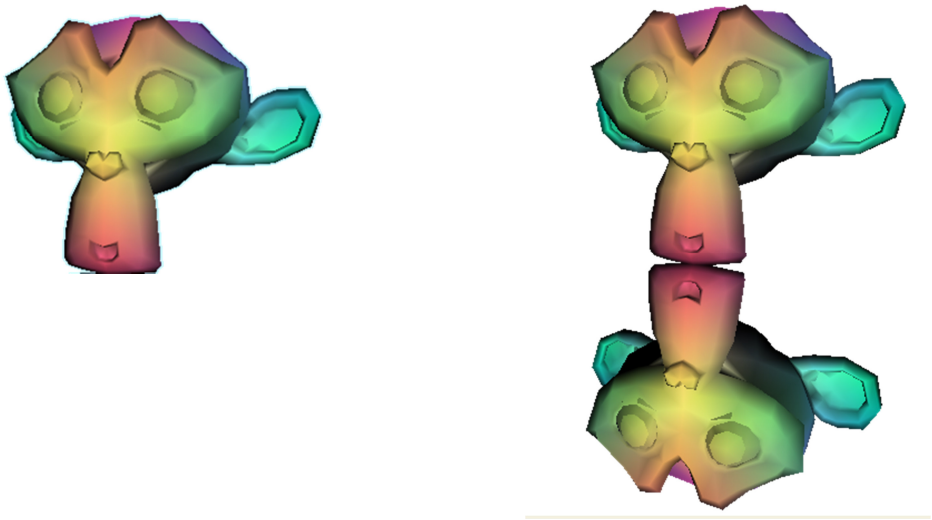

Figura 1. Resultat amb els shaders que us proporcionem (esquerra) y resultat desitjat (dreta).

Vertex shader (no cal modificar-lo):

```
void main()
{
	gl_Position    = gl_Vertex;
	gl_FrontColor  = gl_Color;
}
```

Geometry shader (a completar):

```
void main( void )
{
  for( int i = 0 ; i < 3 ; i++ )
  {
      gl_FrontColor = gl_FrontColorIn[ i ];
      gl_Position  = gl_ModelViewProjectionMatrix  * gl_PositionIn[i];
      EmitVertex();
  }
  EndPrimitive();
  for( int i = 2 ; i >=0 ; i-- )
  {
      gl_FrontColor  = gl_FrontColorIn[ i ];
      vec4 p = gl_PositionIn[i];
      p.y = -p.y;
      gl_Position  = gl_ModelViewProjectionMatrix  * p;
      EmitVertex();
  }
  EndPrimitive();
}
```

Exercici 3 *(1 punt)*

La següent figura mostra un quadrat texturat que ocupa tot un viewport de 1024x1024 pixels. La textura utilitzada és similar a la de la figura. Indica el valor de les següents derivades parcials, on s i t són directament les coordenades de textura interpolades al fragment:

(a) $\frac{\partial s}{\partial x} =$ 1/1024

(b) $\frac{\partial t}{\partial y} =$ 1/1024

Exercici 4 *(1 punt)*

Tenim una aplicació que implementa oclusió ambient i obscuràncies. L'aplicació no fa servir cap estructura de dades per accelerar els càlculs d'intersecció raig-escena.

(a) Què creus que s'executarà més ràpid a l'aplicació, el càlcul de l'oclusió ambient o el càlcul de les obscuràncies?

Ambient occlusion.

(b) Per què?

Donat que no hi ha cap estructura de dades auxiliar, el calcul d'intersecció raig-escena ha de recórrer els objectes **fins trobar una intersecció** (ambient occlusion) o recórrer **tots els objectes** per calcular la intersecció més propera al punt (obscuràncies).

Exercici 5 *(1 punt)*

(a) Quan es pot produir el fenomen de reflexió total?

Quan la llum travessa la superfície que separa un **medi dens** (ex. vidre) d'un **menys dens** (ex. aire)

(b) Quines equacions permeten calcular la proporció de llum reflectida i la proporció de llum transmesa, quan la llum incideix en la superfície de separació entre dos medis?

Les equacions de **Fresnel**

(c) Explica què és l'angle crític
Angle d'incidència a partir del qual no hi ha transmissió, només reflexió.

(d) Indica quin valor de l'angle d'incidència maximitza la proporció de llum transmesa quan aquesta passa de l'aire al vidre.

$\Theta_i = 0$

Exercici 6 *(1 punt)*

Aquest fragment de codi per generar reflexions és incomplet:

```
// 1. Dibuixem el mirall a l'stencil buffer

    glEnable(GL_STENCIL_TEST);
    glStencilFunc(GL_ALWAYS, 1, 1);
    glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE);
    glDepthMask(GL_FALSE);  glColorMask(GL_FALSE...);
    dibuixar(mirall);

// 2. Dibuixem els objectes en posició virtual

    glDepthMask(GL_TRUE); glColorMask(GL_TRUE…);
    glStencilFunc(GL_EQUAL, 1, 1);
    glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);
    glPushMatrix();
    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);
```

(a) Quina crida OpenGL (relacionada amb les matrius de transformació geomètrica) manca?

Cal afegir una crida a **glMultMatrix(matriu simetria**)

(b) On caldria afegir-la (ho pots indicar al codi amb una fletxa)

Immediatament després de glPushMatrix.

Exercici 7 *(1 punt)*

Un company vostre ha fet servir aquest codi per visualitzar l'escena amb VBO.

```
void Object::render() {

        ...
        glBindBuffer(GL_ARRAY_BUFFER, verticesID);
        glBufferData(GL_ARRAY_BUFFER, verts);
        glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, indicesID);
        glBufferData(GL_ELEMENT_ARRAY_BUFFER, indices);

        glBindBuffer(GL_ARRAY_BUFFER, id);
        glVertexPointer(3, GL_FLOAT, 0, (GLvoid*) 0);
        glEnableClientState(GL_VERTEX_ARRAY);

        glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, id);
        glDrawElements(GL_TRIANGLES, n, GL_UNSIGNED_INT, (Glvoid *) 0 );

        glDisableClientState(GL_VERTEX_ARRAY);
        glBindBuffer(GL_ARRAY_BUFFER,0);
        glBindBuffer(GL_ELEMENT_ARRAY_BUFFER,0);

}
```

Tot i que li funciona, no ha aconseguit millorar gens el rendiment respecte els VAs. Què creieu que ha fet malament?

La crida glBufferData s'ha de fer un cop, no cada frame!

Exercici 8 *(1 punt)*

En quins d'aquests casos us poden ser útils les funcions dFdx() i dFdy() de GLSL? Indica-ho amb UTIL / NO UTIL.

(a) Per emular de forma aproximada el funcionament de glPolygonOffset.
(b) Per calcular manualment, al fragment shader, quin nivell de detall (LoD) es necessita quan es fa servir mipmapping.
(c) Per calcular la component especular de la il·luminació.
(d) Per calcular qualsevol derivada parcial en un vertex shader

Exercici 9 *(1 punt)*

Completa aquest codi que implementa ombres per projecció amb stencil buffer:

```
// 1. Dibuixa el receptor al color buffer i al stencil buffer
glEnable(GL_STENCIL_TEST);
glStencilFunc(GL_ALWAYS, 1, 1);
glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE);
dibuixa(receptor);

// 2. Dibuixa oclusor per netejar stencil a les zones a l'ombra
glDisable(GL_DEPTH_TEST);
glColorMask(GL_FALSE, ... GL_FALSE);
glStencilFunc(GL_EQUAL, 1, 1);
glStencilOp(GL_KEEP, GL_KEEP, GL_ZERO);
glPushMatrix(); glMultMatrixf(MatriuProjeccio);
dibuixa(oclusor);
glPopMatrix();
// 3. Dibuixa la part fosca del receptor
glEnable(GL_DEPTH_TEST);
glDepthFunc(GL_LEQUAL);
glColorMask(GL_TRUE, ... , GL_TRUE);
glDisable(GL_LIGHTING);
glStencilFunc(GL_EQUAL, 0, 1);
Dibuixa(receptor);
// 4. Dibuixa l'oclusor
glEnable(GL_LIGHTING);
glDepthFunc(GL_LESS);
glDisable(GL_STENCIL_TEST);
Dibuixa(oclusor);
```

Exercici 10 *(1 punt)*

Indica, per cadascun d'aquests mètodes de filtrat, **quants nivells de mipmap** s'han de consultar per calcular el color de la mostra.

(a) GL_NEAREST 1
(b) GL_LINEAR 1
(c) GL_NEAREST_MIPMAP_NEAREST 1
(d) GL_LINEAR_MIPMAP_LINEAR 2