Laboratorio de Computación Gráfica **24/03/2019**
Moreno Guerra Marco Antonio, Grupo 1
Cuestionario Previo 6, Texturizado

Objetivo: El alumno aplicará las bases de la técnica del mapeo de texturas.

1. **¿Qué es una Textura? En el campo de Gráficas por Computadora.**
   Una textura es un arreglo de datos que contiene información referente al color, la transparencia, normales, profundidades, sombras, entre otras.

   Éstas se aplican sobre píxeles, permitiendo generar patrones visuales complejos que mediante Modelos de Iluminación serían muy caros computacionalmente.

   Una textura es un arreglo de datos, en el que cada elemento de éste se le denomina *Texel* (*Texture Element* o *Texture Pixel*).

   La finalidad de agregar texturas es que permite añadir detalles de superficies por píxel sin necesidad de incrementar la complejidad geométrica de una escena, manteniendo un número bajo de vértices y primitivas. Además, el tiempo de modelado y de renderización se mantiene bajo.

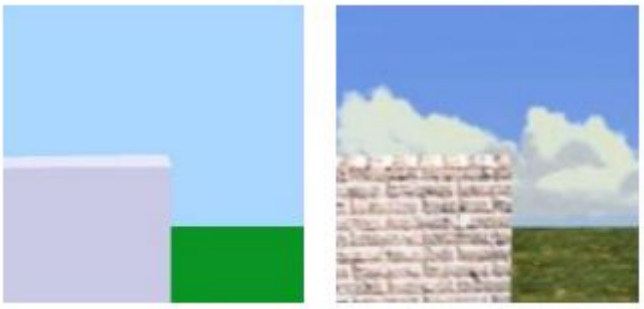

Figura 1. Escena izquierda sin texturas y escena derecha con textura. (Teschner)

Las texturas se pueden clasificar en dos tipos:

- **Generación**
  - ➔ **Archivos almacenados en memoria:** texturas que se encuentran en archivos en un medio de almacenamiento; tales como archivos de extensión .bmp, .jpg, .raw, entre otros formatos de archivos multimedia.
  - ➔ **Procedimentales/Procedurales:** texturas generadas mediante la evaluación de una función o algoritmo, retornando valores de color que pertenecen a los Texels.

- **Tamaño**
  - ➔ **Texturas de una dimensión:** contienen únicamente un Texel de altura y otro de ancho.
  - ➔ **Texturas de dos dimensiones:** texturas con más de un Texel de altura y ancho.
  - ➔ **Texturas de tres dimensiones:** texturas con volumen.

Laboratorio de Computación Gráfica **24/03/2019**
Moreno Guerra Marco Antonio, Grupo 1
Cuestionario Previo 6, Texturizado

2. **¿Qué es el mapeo de texturas?**
   Es la acción de asignar a cada píxel una superficie sus correspondientes Texel de una textura; es decir, es un mapeo, traslación o transformación de coordenadas al aplicar una textura.

   Los Texels se caracterizan por las coordenadas de textura (u,v) en el espacio de textura. Las coordenadas (u,v) son asignadas a un vértice (x,y,z). No es posible hacer una asignación directa de un Texel a un Píxel debido a que la superficie puede cambiar durante la ejecución del programa.

   Las Coordenadas de Texturizado que definirán a la textura (u,v) siempre se encontrarán entre el rango de [0,1].

   El mapeo de textura se aplica, generalmente, por fragmento; mientras que la búsqueda de textura se realiza por fragmento utilizando coordenadas de textura interpoladas.

3. **En texturizado en Gráficos por Computadora, ¿a qué se le llama "Filtro"?**
   Un filtro es un algoritmo que se aplica a las texturas cuando tienen que ser redimensionadas para poder ser aplicadas a una superficie.

   Son de gran utilidad y amplio uso debido a que los objetos virtualizados están en constante cambio de tamaño, por los que se requieren hacer los ajustes de redimensiones para aplicarse en la misma superficie.

4. **¿Qué filtros tiene la versión 2 de OpenGL?**
   GL_NEAREST
   GL_LINEAR
   GL_NEAREST_MIPMAP_NEAREST
   GL_LINEAR_MIPMAP_NEAREST
   GL_NEAREST_MIP

5. **Investigue la manera de construir matemáticamente las siguientes primitivas geométricas, es decir, obtener los vértices que forman a la figura y sus reglas de unión: (no utilizar funciones de GLUT)**

   a) **Prisma, debe recibir como parámetros el alto (eje Y), ancho (eje X) y profundidad (eje Z).**

```
void Mycube(GLfloat length, GLfloat height, GLfloat depth) {
    //base
    glBegin(GL_POLYGON);
    glVertex3f(-length/2, -height/2, depth/2);
    glVertex3f(-length/2, -height/2, -depth/2);
    glVertex3f(length/2, -height/2, -depth/2);
    glVertex3f(length/2, -height/2, depth/2);
    glEnd();
```

```
    // front face
    glBegin(GL_POLYGON);
    glVertex3f(-length/2, -height/2, depth/2);
    glVertex3f(length/2, -height/2, depth/2);
    glVertex3f(length/2, height/2, depth/2);
    glVertex3f(-length/2, height/2 depth/2);
    glEnd();

    // right side face
    glBegin(GL_POLYGON);
    glVertex3f(length/2, -height/2, depth/2);
    glVertex3f(length/2, -height/2, -depth/2);
    glVertex3f(length/2, height/2, -depth/2);
    glVertex3f(length/2, height/2, depth/2);
    glEnd();

    //back side face
    glBegin(GL_POLYGON);
    glVertex3f(-length/2, -height/2, -depth/2);
    glVertex3f(-length/2, height/2, -depth/2);
    glVertex3f(length/2, height/2, -depth/2);
    glVertex3f(length/2, -height/2, -depth/2);
    glEnd();

    //left side face
    glBegin(GL_POLYGON);
    glVertex3f(-length/2, -height/2, depth/2);
    glVertex3f(-length/2, height/2, depth/2);
    glVertex3f(-length/2, height/2, -depth/2);
    glVertex3f(-length/2, -height/Z, -dept/2);
    glEnd();

    //top
    glBegin(GL_POLYGON);
    glVertex3f(-length/2, height/2, depth/2);
    glVertex3f(length/2, height/2, depth/2);
    glVertex3f(length/2, height/2, -depth/2);
    glVertex3f(-length/2, height/2, -depth/2);
    glEnd();
  }
```

b) Cono, debe de recibir como parámetros la altura (eje Y), radio y la resolución de su circunferencia final.

```
void MyCone(GLfloat radius, GLfloat height) {
    GLint i;
    GLdouble theta, ntheta;

    for (i =0; i < circlepoints; i++) {
        glBegin(GL_POLYGON);
        theta =(2*Pl*i/circlegointsl);
        ntheta =(2*Pl*(i + 1)/circle_points);
        glVertex3f(radius*cos(theta), 0, radius*sin(theta));
          gNertex3f(O, height, 0); glVertex3f(radius*cos(ntheta),
0, radius*sin(ntheta));
        glEnd();
```

```
    }
}
```

c) Cilindro, debe de recibir como parámetros la altura (eje Y), radio de las circunferencias y su resolución.

```
void cylinder(float y,float r,int div) {
    int i;

    for(i=1;i<=div;i++) {
        glBegin(GL_TRIANGLES);
        glVertex3f(0,-y/2,0);

glVertex3f(r*cos((i-1)*3.141592/180),-y/2,r*sin((i-1)*3.141592/1
80));

glVertex3f(r*cos(i*3.141592/180),-y/2,r*sin(i*3.141592/180));

        //Plano upper base circunference
        glVertex3f(0,y/2,0);

glVertex3f(r*cos(i*3.141592/180),y/2,r*sin(i*3.141592/180));

glVertex3f(r*cos((i-1)*3.141592/180),y/2,r*sin((i-1)*3.141592/18
0));
        glEnd();

        //Lateral Plane
        glBegin(GL_QUADS);

glVertex3f(r*cos(i*3.141592/180),y/2,r*sin(i*3.141592/180));

glVertex3f(r*cos(i*3.141592/180),-y/2,r*sin(i*3.141592/180));

glVertex3f(r*cos((i-1)*3.141592/180),-y/2,r*sin((i-1)*3.141592/1
80));

glVertex3f(r*cos((i-1)*3.141592/180),y/2,r*sin((i-1)*3.141592/18
0));
        glEnd();
    }
 }
```

6. Indique la diferencia entre un objeto transparente y un objeto translúcido.

Un objeto translúcido es aquel que permite el paso de luz parcialmente a través de sí mismo y el color de su material depende de la luz absorbida, dispersada y reflejada.

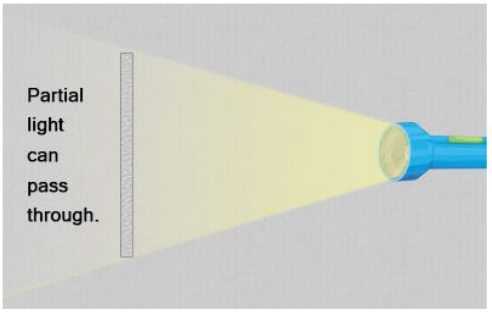

Figura 2. Representación de un objeto traslúcido. (Patkar, 2018)

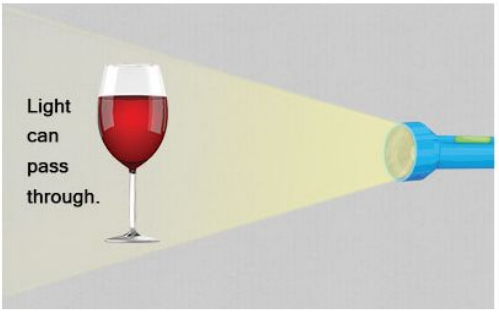

Figura 3. Representación de un objeto transparente. (Patkar, 2018)

Mientras que un objeto transparente es aquel que permite el paso completo de la luz y el color del material depende de la luz que emite.

Conclusiones

Es importante conocer y aprender el uso de las texturas en la computación gráfica, pues se trata de una herramienta clave para incrementar el realismo de los modelos virtualizados, sin necesidad de aumentar la complejidad matemática o algorítmica del código. Esto implica el practicar y conocer la forma de mapear los Texels a los píxeles para obtener buenos resultados; así como el uso de filtros para adaptar las texturas a virtualizaciones dinámicas, donde se mueve constantemente la vista o cambian de dimensiones los objetos por la proyección perspectiva.

Referencias

(s.f.). Obtenido de http://luissergiov.sodvi.com/alexsoto/Texturizado01a.pptx

Govil-Pai, S. (2004). *Principles of Computer Graphics Theory and Practice Using OpenGL and Maya.* California, United States Of America: Springer.

Patkar, P. (8 de mayo de 2018). *Science Struck*. Obtenido de https://sciencestruck.com/difference-between-translucent-transparent-opaque-material

Teschner, M. (s.f.). *University of Freiburg.* Obtenido de Computer Science Department: https://cg.informatik.uni-freibur