Facultad de Informática

UNIVERSIDAD DE MURCIA

AUTÓMATAS Y LENGUAJES FORMALES

GRADO EN INGENIERÍA INFORMÁTICA

Práctica 2 - Curso 2019/2020

Autores:
Eduardo MARTÍNEZ GRACIÁ
edumart@um.es
Mercedes VALDÉS VELA
mdvaldes@um.es
Santiago PAREDES MORENO
chapu@um.es
José Manuel JUÁREZ HERRERO
jmjuarez@um.es

5 de noviembre de 2019

Introducción

El objetivo de esta práctica es desarrollar un analizador de ficheros en formato OBJ en el lenguaje de programación Python haciendo uso de expresiones regulares. El formato OBJ, definido por la empresa *Wavefront Technologies*, se emplea para representar objetos tridimensionales que pueden ser visualizados mediante un motor gráfico o pueden imprimirse mediante impresoras 3D. Una vez verificada la corrección del fichero de entrada, el objeto cargado se visualizará mediante un visor que emplea la librería Glumpy de Python.

Los ficheros OBJ son ficheros de tipo texto en formato ASCII, con extensión `.obj`. En esta práctica sólo validaremos parcialmente la información contenida en los ficheros OBJ, limitándonos a la descripción de la superficie geométrica de objetos descritos mediante triángulos. El formato OBJ permite representar otra información, como polígonos complejos, colores, texturas, materiales, etc. Esas características no serán analizadas en esta práctica.

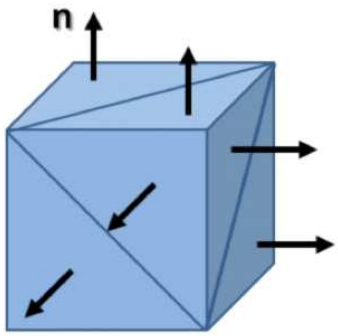

Figura 1: Representación de un cubo formado por 12 triángulos

Formato OBJ

La figura 1 muestra un cubo definido por un total de 12 triángulos. Cada triángulo se puede definir mediante los tres puntos que especifican sus vértices. Para que el motor 3D pueda calcular el efecto de la iluminación sobre las superficies, también es importante indicar el vector normal que define la parte exterior del triángulo. En la figura 1 aparece indicado un vector normal por cada triángulo. El sentido del vector especifica la cara externa. Los ficheros OBJ permiten que cada vértice tenga su propio vector normal, interpolando los vectores para cualquier punto de una superficie mediante los vectores de sus vértices. De esta forma se puede lograr una visualización más realista.

Especificación de coordenadas

En los ficheros OBJ, las coordenadas de vértices y vectores se especifican mediante valores en punto flotante. En esta práctica se debe comprobar que los valores en punto flotante cumplen las siguientes características:

- El número flotante sólo comienza con signo cuando éste es negativo. Si es positivo, no se indica el signo.

- La mantisa del número flotante está formada por uno o más dígitos correspondientes a la parte entera, seguidos opcionalmente por la parte decimal formada por el carácter `.` y una secuencia de entre uno y siete dígitos.
- El exponente es opcional. En caso de que se indique, debe comenzar con `e` o con `E`. Tras este carácter, siempre se indica el signo del exponente, `+` o `-`. En el caso de que el exponente sea 0 se debe emplear el signo `+`. El valor del exponente debe indicarse con un mínimo de un dígito.
- Se pueden usar dígitos `0` no significativos tanto en la mantisa como en el exponente.

Ejemplos de números válidos son:

- 20.6480e-01
- -0.123
- 0.0E+00

Ejemplos de números no válidos son:

- 50.e-01
- .765
- 0.01E-00

Comandos gráficos

Cada línea del fichero OBJ contiene una sentencia que define una parte de la geometría del objeto. Las líneas que nos interesan en esta práctica comienzan por los comandos `v`, `vn`, `vt` o `f`. Cualquier otra línea no es relevante para esta práctica y puede ser ignorada.

Especificación de vértices

Las líneas del fichero OBJ que especifican los vértices empiezan por el comando `v`, que va seguida de hasta cuatro valores en punto flotante separados entre sí por uno o más espacios en blanco:

```
1 v x y z w
```

Los tres primeros valores flotantes especifican las coordenadas `(x,y,z)` del vértice. El formato OBJ permite especificar ciertas curvas y superficies que requieren una cuarta coordenada llamada *peso*. El programa de la práctica debe comprobar que los vértices del fichero se especifican con tres o cuatro coordenadas. En el caso de usar cuatro, el peso debe tomar siempre el valor `1.0`.

Conforme aparecen vértices en el fichero OBJ, estos reciben implícitamente un identificador numérico. El primer vértice tiene como identificador el 1, y los restantes vértices reciben como identificador el número correspondiente al orden de aparición del vértice.

Especificación de vectores normales

Un segundo tipo de línea que se debe verificar en el fichero OBJ es el que permite especificar un vector normal. Este tipo de línea comienza con el comando `vn` y va seguido de tres valores flotantes `(i,j,k)` que especifican el vector, separados entre sí por uno o más espacios en blanco:

```
1 vn i j k
```

Al igual que sucede con los vértices, conforme aparecen vectores normales en el fichero OBJ, estos reciben implícitamente un identificador numérico. El identificador comienza en 1 y corresponde al orden de aparición del vector. El espacio de índices de los vectores normales es distinto del de los vértices (se distinguen por el contexto en el que se usan).

Especificación de vértices de textura

Un tercer tipo de línea, iniciada por el comando `vt`, permite indicar un vértice de una textura. Una textura puede ser bidimensional - por ejemplo, una imagen JPG o PNG - cuando se debe visualizar sobre una superficie plana, o bien puede ser tridimensional si se tiene que visualizar sobre una superficie con relieve. Estas coordenadas se asocian a vértices definidos con el comando `v` para superponer la textura sobre la superficie, como se indica posteriormente.

El comando `vt` va seguido de dos o tres valores flotantes que especifican coordenadas de la textura, separados entre sí por uno o más espacios en blanco:

```
1 vt u v w
```

La coordenada `w` es opcional, mientras que `u` y `v` son obligatorias. Se debe comprobar que todos los vértices de textura usen la misma cantidad de coordenadas (dos o tres).

Conforme aparecen vértices de texturas, estos reciben implícitamente un identificador numérico. El identificador comienza en 1 y corresponde al orden de aparición del vértice. El espacio de índices de los vértices de textura es distinto del de los vértices y los vectores normales (se distinguen por el contexto en el que se usan).

Especificación de caras del objeto

Por último, el programa debe analizar las líneas que especifican caras (faces) del objeto. Estas líneas comienzan con el comando `f` y van seguidas de una lista de tripletas separadas por espacios en blanco.

Cada tripleta está formada por tres índices separados por el carácter `/` :

- El primer índice referencia a un vértice previamente especificado mediante un comando `v`.
- El segundo índice referencia a un vértice de textura previamente especificado mediante un comando `vt`.
- El tercer índice referencia a un vector normal previamente especificado mediante un comando `vn`.

Como se ha indicado previamente, el primer índice toma el valor 1. No debe haber ningún espacio en blanco entre los números y los separadores / :

```
f v/vt/vn v/vt/vn v/vt/vn
```

El programa debe comprobar que cada cara especifica un triángulo, es decir, tiene exactamente tres tripletas. El formato OBJ permite que se definan caras con más de tres vértices, pudiendo forman polígonos más complejos. No obstante, en esta práctica sólo se permitirá la definición de caras con tres vértices.

En las tripletas, el único elemento obligatorio es el primer índice, que especifica el vértice. Los demás son opcionales, pero todas las tripletas deben ser consistentes en la especificación de la información, es decir, si la primera tripleta especifica el vértice de textura o el vector normal, las dos restantes también lo deben hacer.

Cuando una tripleta sólo contiene el índice del vértice, es opcional usar los separadores /. Cuando no contiene el vector normal, es opcional usar el segundo separador /. Cuando no contiene el vértice de textura, sí se tienen que usar los dos separadores /.

Suponiendo que existan los vértices y vectores correspondientes, consideraremos válidas las siguientes especificaciones de caras:

```
f 1 2 3
f 1/ 2/ 3/
f 1// 2// 3//
f 1/1 2/2 3/3
f 1/1/ 2/2/ 3/3/
f 1//1 2//2 3//3
```

No son válidas las siguientes especificaciones:

```
f /1/ /2/ /3/
f 1/1/1 2/2/2 3//3
f 1/1 2/2 3/3 4/4
```

Visualización del objeto

Preparación de los datos

Para la visualización de los datos, puedes encontrar en la carpeta `Recursos > Prácticas > Práctica2` del sitio de la asignatura en el Aula Virtual un módulo Python llamado `visor.py` que contiene la función:

```
def mostrar(V,T,N,F_V,F_T,F_N)
```

Esta función recibe los siguientes argumentos:

- `V`: lista de coordenadas de los vértices del objeto. Para rellenar esta lista, hay que añadir cada vértice en el orden en que aparece en el fichero. Por cada vértice, añade a V una lista con las tres coordenadas `[x,y,z]` en orden. No insertes el valor `w` (peso).

- `T`: lista de coordenadas de los vértices de textura del objeto. Para rellenar esta lista, hay que añadir cada vértice de textura en el orden en que aparece en el fichero. Por cada vértice, añade a T una lista con las dos o tres coordenadas `[u,v,w]` en orden.
- `N`: lista de coordenadas de los vectores normales del objeto. Para rellenar esta lista, hay que añadir cada vector en el orden en que aparece en el fichero. Por cada vector, añade a N una lista con las tres coordenadas `[i,j,k]` en orden.
- `F_V`: lista de valores `int` con los índices de los vértices de las caras del objeto, donde el primer índice comienza en 0. Para rellenar esta lista, hay que añadir cada índice en el orden en el que aparecen las caras en el fichero. Por cada cara, añade los tres índices de los vértices del triángulo en orden.
- `F_T`: lista de valores `int` con los índices de los vértices de textura del objeto, donde el primer índice comienza en 0. Para rellenar esta lista, hay que añadir cada índice en el orden en el que aparecen las caras en el fichero. Por cada cara, añade los tres índices de los vértices de textura del triángulo en orden (en caso de que aparezcan).
- `F_N`: lista de valores `int` con los índices de los vectores normales de las caras del objeto, donde el primer índice comienza en 0. Para rellenar esta lista, hay que añadir cada índice en el orden en el que aparecen las caras en el fichero. Por cada cara, añade los tres índices de los vectores normales en orden (en caso de que aparezcan).

Para que funcione correctamente la visualización, tienes que instalar la librería `glumpy` de Python, como se indica más adelante.

En la carpeta `Recursos > Prácticas > Práctica2` puedes encontrar unos cuantos ficheros OBJ de ejemplo.

Cálculo de vectores normales

En el caso de que una cara del fichero OBJ no especifique vectores normales para sus vértices, será necesario calcular el vector normal del triángulo e insertar la información necesaria en las listas N y F_N que se pasan al método `mostrar()` de `visor.py`.

La figura 2 indica cómo se puede calcular el vector normal de una cara. Habría que asignar el mismo vector normal a los tres vértices de las caras que no tienen vectores normales.

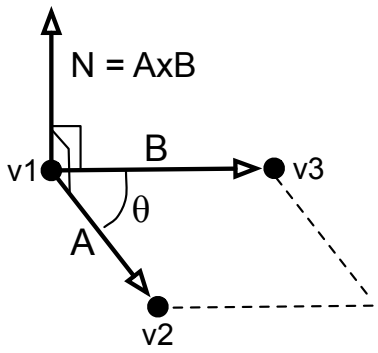

Figura 2: Cálculo del vector normal

Supongamos que los vértices de un triángulo son $v1$, $v2$ y $v3$. Con estos vértices podemos calcular los vectores A y B:

$$A = (A_x, A_y, A_z) = (v2_x - v1_x, v2_y - v1_y, v2_z - v1_z)$$
$$B = (B_x, B_y, B_z) = (v3_x - v1_x, v3_y - v1_y, v3_z - v1_z)$$

El vector normal $N = A \times B$ se puede obtener como:

$$N = (N_x, N_y, N_z) = A \times B = (A_y B_z - A_z B_y, A_z B_x - A_x B_z, A_x B_y - A_y B_x)$$

El vector normal debe tener módulo 1. Para ello, hay que realizar los siguientes cálculos:

$$|N| = \sqrt{N_x^2 + N_y^2 + N_z^2}$$
$$n = \left(\frac{N_x}{|N|}, \frac{N_y}{|N|}, \frac{N_z}{|N|} \right)$$

El vector normal obtenido es n.

Glumpy

Linux o Mac OS X

La instalación en Linux o Mac OS X requiere los siguientes paquetes:

```
pip install numpy
pip install cython
pip install pyopengl
pip install triangle
pip install glumpy
pip install pyglet
```

Los cuatro primeros paquetes son dependencias de `glumpy`. El paquete `pyglet` es una librería de OpenGL, el sistema gráfico usado por `glumpy`.

Si tienes la versión 2.7 de Python en tu equipo, es posible que en lugar de `pip` tengas que usar `pip3` para que lo anterior esté diponible para la versión 3 de Python.

Windows

La instalación en Windows es parecida a la de Linux o Mac OS X al comienzo. Tienes que abrir una ventana de comandos y ejecutar:

```
pip install numpy
pip install cython
pip install pyopengl
pip install triangle
pip install glumpy
```

Si lo anterior falla, posiblemente se deba a que es necesario instalar previamente las herramientas de Visual Studio para compilar C/C++ (muchas librerías de Python tienen una parte implementada en estos lenguajes).

Después de instalar los paquetes indicados, sigue las instrucciones especificadas en la página `http://glumpy.readthedocs.io/en/latest/installation.html` para descargar dos librerías dinámicas `freetype.dll` y `glfw.dll` e instalarlas en tu equipo.

Trabajo a realizar

Crea un módulo `practica2.py` que implemente un analizador y visor de ficheros OBJ:

1. Añade un código principal a `practica2.py`. Dicho código debe solicitar al usuario, a través de la consola, el nombre de un fichero OBJ a analizar. El nombre del fichero debe terminar con la extensión `.obj`. Si el fichero no existe, o no tiene la extensión `.obj`, el programa debe volver a solicitar al usuario el nombre de un fichero. El usuario puede terminar el programa introduciendo una cadena vacía.
2. La verificación del formato OBJ debe cubrir los aspectos indicados en las secciones anteriores, es decir, se debe verificar que la especificación del objeto cumple el formato indicado en las líneas que empiezan por `v`, `vt`, `vn` y `f`.
3. Cualquier error detectado debe ser informado con claridad a través de la salida de errores de la aplicación. El mensaje debe comenzar con el número de línea en la que se ha detectado el error. Cualquier error provocará la finalización del programa.
4. La visualización del objeto se realizará al final de la verificación del fichero OBJ completo.
5. Para la verificación del formato y comprobación de extensión del fichero, se emplearán los métodos del paquete `regex`. No se puede emplear para la verificación del formato ningún método de cadenas de caracteres de Python, como troceado o `split()`.
6. El código de `practica2.py` debe estar estructurado en métodos. Se penalizará el uso de métodos muy largos, mal estructurados, mal documentados o con nombres de variables poco significativas.
7. Sólo se podrá verificar el fichero OBJ realizando un único recorrido de comienzo a fin.

Evaluación

En caso de que se cumplan todos los requisitos indicados, esta práctica se evaluará con 3 puntos.