Facultad de Informática

UNIVERSIDAD DE MURCIA

PRÁCTICA 2

Autómatas y Lenguajes Formales

Última actualización: 5 de noviembre de 2019

Eduardo Martínez Graciá, Mercedes Valdés Vela,
Santiago Paredes Moreno, José Manuel Juárez Herrero

Dpto. de Ingeniería de la Información y las Comunicaciones
Facultad de Informática de la **Universidad de Murcia**

INDICE

INTRODUCCIÓN

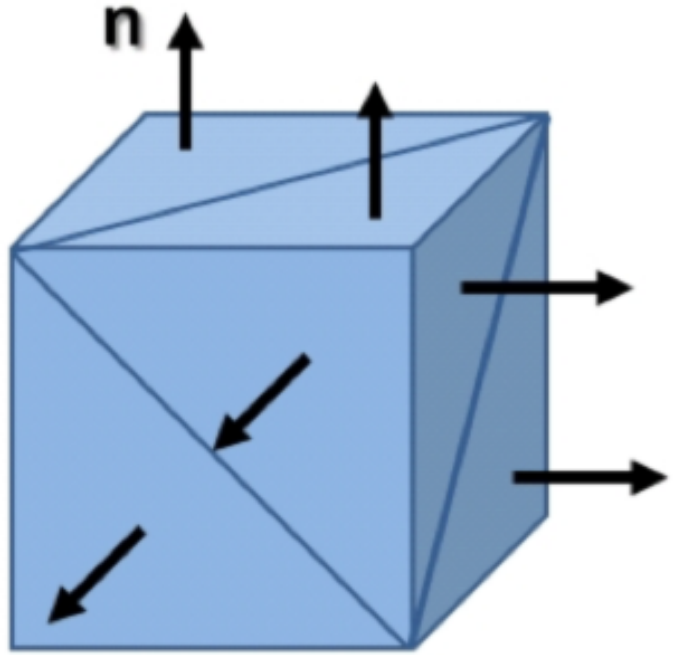
INTRODUCCIÓN

- Objetivo: analizar y visualizar ficheros en formato OBJ.
- Son ficheros ASCII con extensión `.obj` para representar objetos tridimensionales.
- Usaremos `regex` para analizar y extraer información.
- Usaremos `glumpy` para visualizar la información a través del módulo `visor.py` proporcionado.

FORMATO OBJ

FORMATO OBJ

- Definido inicialmente por *Wavefront Technologies*.
- Analizamos parcialmente el fichero OBJ (el formato permite mucho más).
- Un objeto 3D se define mediante múltiples triángulos.
- Cada triángulo se especifica mediante tres vértices.



  - Mediante vectores normales se indica la parte externa de la superficie (para efectos de iluminación).

COORDENADAS (1)

Las coordenadas de vértices y vectores se especifican mediante valores en punto flotante:

- Sólo comienza con signo cuando éste es negativo.
- La mantisa 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. Debe comenzar con `e` o `E`, 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.

COORDENADAS (2)

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

COMANDOS GRÁFICOS

- Las líneas del fichero OBJ contienen sentencias gráficas.
- Nos interesan las que comienzan por los comandos:
  - `v`: define un vértice
  - `vn`: define un vector normal
  - `vt`: define un vértice de textura
  - `f`: define un triángulo (facet)
- Cualquier otra línea no es relevante para la práctica y puede ser ignorada.
- Cada vértice tiene un identificador numérico implícito que comienza en 1. Lo mismo sucede con los vectores normales y vértices de textura. Los identificadores sirven para referenciar estas tres entidades desde los comandos `f`.

ESPECIFICACIÓN DE VÉRTICES

- Las líneas del fichero OBJ que especifican los vértices empiezan por `v` y tienen 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 cuarto valor `w` (peso) es opcional y, en caso de aparecer, debe tener un valor equivalente a `1.0`.

ESPECIFICACIÓN DE VECTORES NORMALES

- Las líneas del fichero OBJ que especifican los vectores normales empiezan por `vn` y tienen tres valores en punto flotante separados entre sí por uno o más espacios en blanco:

```
1 vn i j k
```

- Los tres valores flotantes especifican las coordenadas `[i,j,k]` del vector.

ESPECIFICACIÓN DE VÉRTICES DE TEXTURAS

- Una textura bidimensional puede ser una imagen JPG o PNG que se visualiza sobre una superficie plana.
- Mediante coordenadas (valores flotantes) se especifican puntos de la textura que se superponen sobre vértices de la superficie, como se indica posteriormente.
- Las líneas del fichero OBJ que especifican los vértices de texturas empiezan por `vt` y tienen tres valores en punto flotante separados entre sí por uno o más espacios en blanco:

```
1  vt  u  v  w
```

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

ESPECIFICACIÓN DE CARAS DEL OBJETO (1)

- Las caras de los objetos comienzan con `f` (facet) 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`.
- No debe haber ningún espacio en blanco entre los números y los separadores `/` :

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

ESPECIFICACIÓN DE CARAS DEL OBJETO (2)

- El programa debe comprobar que cada cara especifica un triángulo, es decir, tiene exactamente tres tripletas.
- En las tripletas, el único elemento obligatorio es el primer índice, que especifica el vértice.
- Todas las tripletas de una cara 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 `/`.

ESPECIFICACIÓN DE CARAS DEL OBJETO (3)

- Especificaciones válidas:

```
1 f 1 2 3
2 f 1/ 2/ 3/
3 f 1// 2// 3//
4 f 1/1 2/2 3/3
5 f 1/1/ 2/2/ 3/3/
6 f 1//1 2//2 3//3
```

- Especificaciones no válidas:

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

VISUALIZACIÓN DEL OBJETO

VISOR.PY

Usa `Recursos > Prácticas > Práctica2 > visor.py`:

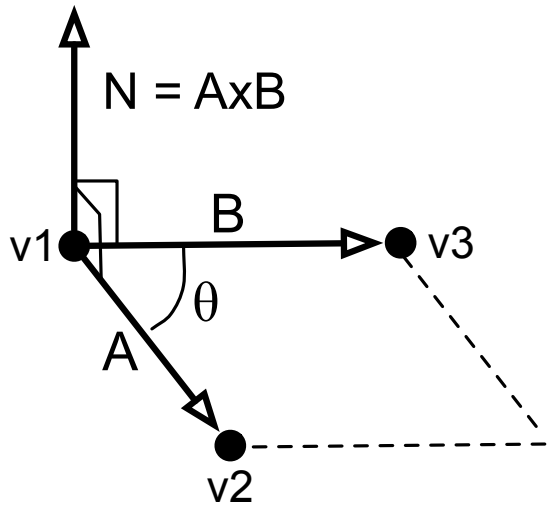
```
1 def mostrar(V,T,N,F_V,F_T,F_N)
```

- `V`: vértices. Por cada uno, añade `[x,y,z]`.
- `T`: vértices de textura. Por cada uno, añade `[u,v,w]`.
- `N`: vectores normales. Por cada uno, añade `[i,j,k]`.
- `F_V`: índices de los vértices. Por cada cara, añade los tres índices de los vértices.
- `F_T`: índices de los vértices de textura. Por cada cara, añade los tres índices de los vértices de textura.
- `F_N`: índices de los vectores normales. Por cada cara, añade los tres índices de los vectores normales.

ATENCIÓN: para Python, el primer índice comienza en 0.

CÁLCULO DE VECTORES NORMALES (1)

- Si una cara no especifica vectores normales para sus vértices, será necesario calcularlos.
- Habría que asignar el mismo vector normal a los tres vértices de las caras.



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

$$A = (A_x, A_y, A_z) = (v2_x - v1_x, v2_y - v1_y, v2_z - v1_z)$$

$$B = (B_x, B_y, B_z) = (v3_x - v1_x, v3_y - v1_y, v3_z - v1_z)$$

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

CÁLCULO DE VECTORES NORMALES (2)

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 del triángulo y los índices se insertan en las listas `N` y `F_N` que se pasan al método `mostrar()` de `visor.py`.

INSTALACIÓN DE GLUMPY

LINUX O MAC OS X

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

```
1 pip install numpy
2 pip install cython
3 pip install pyopengl
4 pip install triangle
5 pip install glumpy
6 pip install pyglet
```

Los cuatro primeros paquetes son dependencias de `glumpy`. El paquete `pyglet` es una librería de OpenGL.

WINDOWS

La instalación en Windows requiere:

```
1 pip install numpy
2 pip install cython
3 pip install pyopengl
4 pip install triangle
5 pip install glumpy
```

Es necesario instalar previamente las herramientas de Visual Studio para compilar C/C++.

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

TRABAJO Y EVALUACIÓN

TRABAJO

Implementa un módulo `practica2.py` para:

1. Pedir iterativamente un fichero OBJ (comprueba la extensión y que se pueda abrir). Termina si es cadena vacía.
2. Verifica el formato indicado en las líneas `v`, `vt`, `vn` y `f`.
3. Cualquier error detectado debe ser informado con claridad a través de la salida de errores de la aplicación.
4. Extrae la información para visualizar el objeto al final de la verificación del fichero OBJ completo.
5. Sólo se podrá realizar un recorrido del fichero OBJ para verificarlo.

Usa los métodos del paquete `regex`.
No se puede emplear ningún método de cadenas de caracteres de Python, como comparación, troceado o `split()`.

EVALUACIÓN

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