Informática Gráfica II

Object-oriented Graphics Rendering Engine

Práctica 1.1: Los laberintos

Alberto Núñez
Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid

Objetivo

- La primera práctica consiste en desarrollar un juego similar al clásico Pac-Man
- El objetivo principal es utilizar el motor Ogre3D para renderizar escenas
- La lógica del juego es muy sencilla, y la usaremos para unir los elementos de la escena
  - El héroe (Sinbad)
  - Los villanos
    - Ogrehead
    - Un villano que crearemos nosotros
  - Luces
  - Cámaras
  - Texturas
  - Shadders

Estructura del laberinto

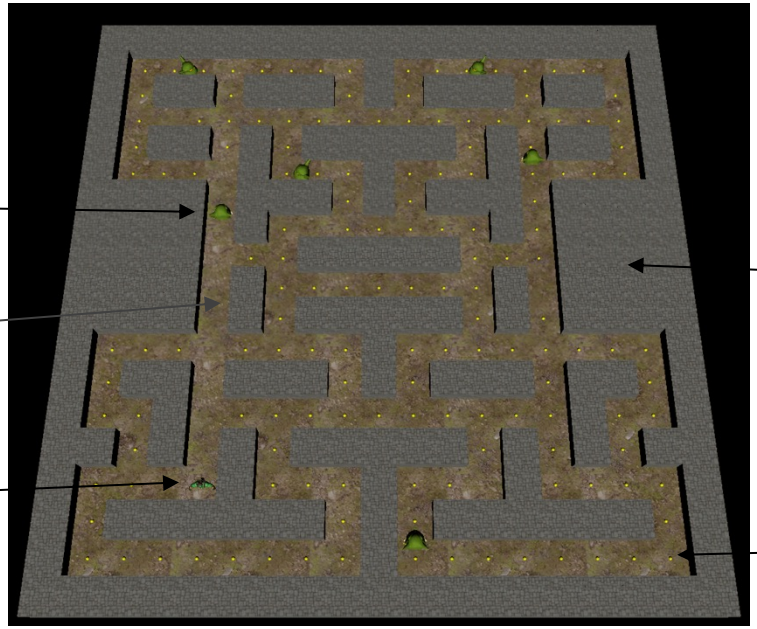
- La primera parte que vamos a construir es el laberinto del juego
- ~~Una opción es codificar la estructura del laberinto en una clase del programa~~
  - Esto nos obliga a recompilar cada vez que cambiemos el laberinto ☹
- Otra opción es codificar el laberinto en un fichero
  - No es necesario recompilar
  - Podemos crear laberintos fácilmente

Estructura del laberinto

- Los laberintos se representan como una cuadrícula
  - Cada posición es un cubo. Tenemos dos tipos:
    - Muro (sólido y no traspasable)
    - Hueco (camino por donde se mueve el héroe)
    - En cada hueco hay una perla
      - Cuando el héroe coge todas las perlas, gana!

Estructura del laberinto

- Los laberintos se representan como una cuadrícula
  - Cada posición es un cubo. Tenemos dos tipos:
    - Muro (sólido y no traspasable)
    - Hueco (camino por donde se mueve el héroe)
    - En cada hueco hay una perla
      - Cuando el héroe coge todas las perlas, gana!



Estructura del laberinto

- El fichero que tendrá la información tiene el siguiente formato

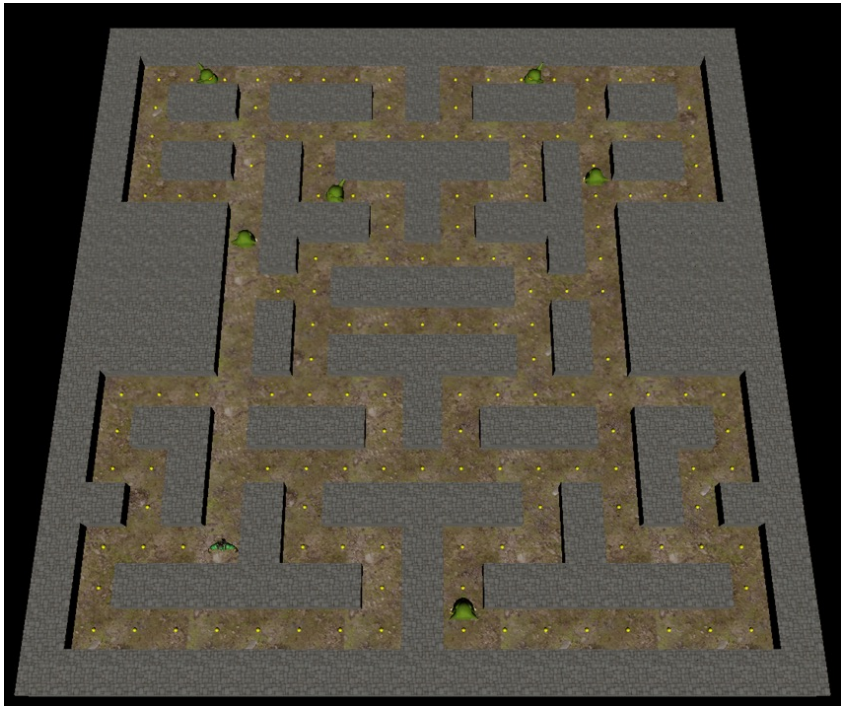
```
NumFilas
NumColumnas
caracteresFila_1
caracteresFila_2
…
caracteresFila_NumFilas
```

- **NumFilas** representa el número de filas del laberinto
- **NumColumnas** representa el número de columnas del laberinto
- **caracteresFila_i** representa la secuencia de caracteres de la fila i-ésima
  - Cada carácter puede tener el valor 'x' (muro) y 'o' (perla)

Estructura del laberinto

- Ejemplo de laberinto codificado en el fichero
  - 19 filas
  - 19 columnas

```
19
19
xxxxxxxxxxxxxxxxxxx
xooooooooxoooooooox
xoxxoxxxoxoxxxoxxox
xooooooooooooooooox
xoxxoxoxxxxxoxoxxox
xooooxoooxoooxoooox
xxxxoxxxoxoxxxoxxxx
xxxxoxooooooooxxxxx
xxxxoooxxxxxoooxxxx
xxxxoxooooooooxxxxx
xxxxoxoxxxxxoxoxxxx
xooooooooxoooooooox
xoxxoxxxoxoxxxoxxox
xooxoooooooooooxoox
xxoxoxoxxxxxoxoxoxx
xooooxoooxoooxoooox
xoxxxxxxoxoxxxxxxox
xooooooooxoooooooox
xxxxxxxxxxxxxxxxxxx
```



Implementación

- Tendremos dos tipos de bloque
  - Muro
  - Hueco (contiene una perla)
- Se diferencian en dos aspectos
  - Uno es traspasable, el otro no.
  - Cada uno tiene una entidad
- Tiene sentido crear una clase para cada tipo de bloque
- Ambos heredan de `IG2Object`
- El laberinto es una secuencia de bloques
- También tiene sentido crear una clase para el laberinto
  - Contiene los bloques
  - Contiene la lógica para crear el laberinto a partir del fichero