Informática Gráfica II

Object-oriented Graphics Rendering Engine

Práctica 1.3: Los villanos

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

Objetivo

- En este apartado vamos a diseñar un villano y a implementar sus movimientos
- Tendremos dos tipos de villanos
  - Uno de diseño propio
  - El resto que usará la malla de *ogrehead*
- La posición inicial de los villanos se indicará en el fichero del laberinto
  - Utilizaremos el carácter 'v' (minúscula) para indicar la posición de un villano (ogrehead)
  - Podremos utilizar 'V' (mayúscula) para indicar la posición del villano diseñado
  - La lógica de ambos puede ser la misma
- Se valorará especialmente la calidad del código implementado

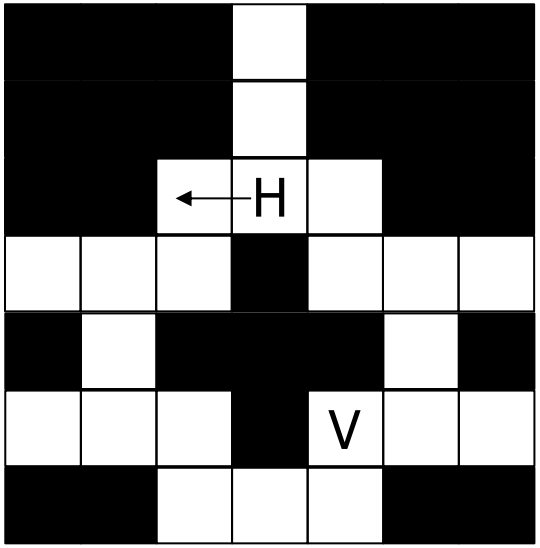
Villano de diseño propio

- Se creará un villano utilizando las mallas proporcionadas por Ogre3D
- En particular, debe cumplir las siguientes condiciones:
  - Estar formado por, al menos, **tres mallas distintas**.
  - Estar formado por, al menos, **diez entidades**.
  - Contener, al menos:
    - **Dos partes móviles** que tengan, al menos, **tres entidades** cada una que **realicen rotaciones**.
    - **Un *timer*** que controle el tiempo que las partes móviles realizarán movimientos de rotación en cada sentido.
- Se valorará un buen diseño de las clases que formen el villano. Por ejemplo, que:
  - Sea parametrizable
  - Evite Código repetido
  - Sea reutilizable

Movimientos de los villanos

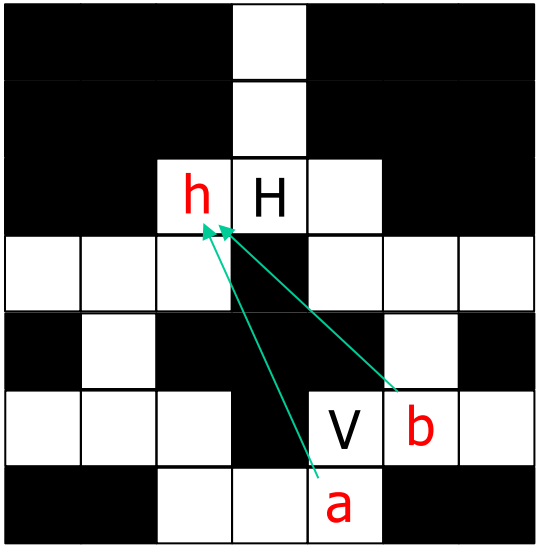
- La lógica del villano debe cumplir los siguientes requisitos
  - Un *villano* nunca cambia de sentido (girar 180 grados)
    - A menos que sea su única opción para no quedarse bloqueado.
  - Un *villano* sólo calcula una nueva dirección en los siguientes casos:
    - 1) Está bloqueado.
    - 2) Está en una posición donde es posible realizar un giro de 90 grados y avanzar.
      - La dirección tomada será la que minimice la distancia euclídea entre el *héroe* y el centro del primer bloque visitado por el *villano*.
  - En cada paso que dé el *villano* (si no está bloqueado) avanzará una unidad en la posición correspondiente.
- Adicionalmente, se puede emplear el uso de *timers* para definer estados. Por ejemplo:
  - Tiempo de ataque: Se persigue al héroe
  - Tiempo de huida: Se huye del héroe

Movimientos de los villanos (Ejemplo 1)



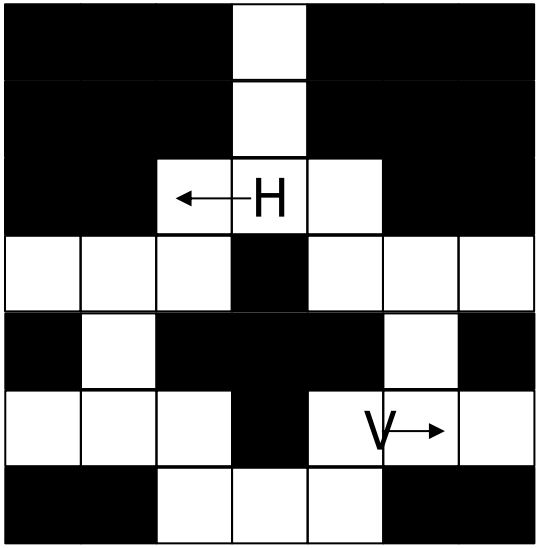
- Supongamos el siguiente escenario
  - El héroe (H)
  - El villano (V)
  - Flechas indican movimiento actual
- El el siguiente update:
  - El héroe sigue su movimiento ←
  - ¿Qué hace el villano?
- ¿Está en un cruce?
  - Sí!

Movimientos de los villanos (Ejemplo 1)



- Supongamos el siguiente escenario
  - El héroe (H)
  - El villano (V)
  - Flechas indican movimiento actual
- El el siguiente update:
  - El héroe sigue su movimiento ←
  - ¿Qué hace el villano?
- ¿Está en un cruce?
  - Sí!
- Calculamos distancias (euclídeas) entre
  - a y h
  - b y h
- Utiliza la dirección que minimice la distancia
  - Siguiente bloque visitado
  - Bloque de destino del héroe

Movimientos de los villanos (Ejemplo 2)



- Supongamos el siguiente escenario
  - El héroe (H)
  - El villano (V)
  - Flechas indican movimiento actual
- El el siguiente update:
  - El héroe sigue su movimiento ←
  - ¿Qué hace el villano?
- ¿Está en un cruce?
  - **No!**
  - **No se calcula nueva dirección!**

El Timer

- Opcionalmente, se puede usar un *timer* para definir estados
- Su uso es sencillo. Se debe incluir:

```
#include <OgreTimer.h>
```

- Se define con:

```
Ogre::Timer* timer;
```

- Se reinicia con:

```
timer->reset();
```

- El tiempo transcurrido (en ms) se obtiene con:

```
timer->getMilliseconds();
```