

# Manual de la librería “ObjAnotado.h”

## 1 Introducción

La librería de ‘ObjAnotado.h’ proviene de la necesidad que tuve en mi próxima librería ‘Automata.h’ de indicar el estado del automata (o al menos algunos estados) en la listas de objetos presentes en la localidad o bien transportados por el jugador. Posteriormente mientras ampliaba mi también futura ‘Moviles’, a modo de generalización de ‘PNJMovil’, para utilizar con mis autómatas; Zak me puso en contacto con Jaevius y su librería de puertas realistas.

Su librería de puertas estaba ‘pidiendo a voces’ que las puertas indicasen si estaban abiertas o cerradas en la lista de objetos, además en el desarrollo de ‘Moviles’ yo me había topado con la necesidad de incluir rastros que se pudiesen seguir lo que a su vez devenía en el desarrollo de una nueva librería ‘Rastros’. De nuevo mis ‘rastros’ agradecerían el haber incluido en su código la posibilidad de indicar (opcionalmente) hacia dónde se dirigían o incluso si había varios rastros similares que se encaminaban en direcciones diferentes. De nuevo el objeto requería poder modificar su descripción en la lista de objetos presentes o transportados.

Existe una posibilidad ‘standard’ de hacer esto, modificar la definición de ‘nombre\_corto’, pero esta posibilidad tiene la desagradable consecuencia de que en TODAS partes y no sólo en los listados, aparecería la ‘anotación’ del objeto. Así pongamos que cambiamos el ‘nombre\_corto’ de, digamos, un gato para indicar que se encuentra ‘muy asustado’. En el listado aparecería correctamente:

```
Puedes ver un gato (muy asustado)
```

Pero con el problema de que tras la orden ‘lanzar gato’, obtendrías algo así como:

```
(primero coges el gato (muy asustado))  
No serviría de nada.
```

Que no digo yo que en este caso no sea adecuado, por eso de que el gato te está viendo las intenciones de usarlo cual pelota de rugby, pero que en general no parece lo más adecuado.

Así que la nueva librería modifica la forma en la que se listan los objetos y permite a algunos de ellos proporcionar un texto que agregar (entre paréntesis) a su descripción corta.

## 2 Inicios básicos: ¿Cómo usarla?

Para poder usar la librería de anotaciones es necesario añadir dos includes en tu código de InformatE. El primero debe estar antes de el include de Acciones, consiste en la sustitución de la rutina inicial de la librería para escribir 'elementos' tras el nombre del objeto en un listado:

```
include "NueETE";  
include "Acciones";
```

el otro include debe estar detrás del include de la gramática. Mis futuras librerías 'Rastros' y 'Automatas' la van a necesitar así que además deberá estar ANTES de estas para que todo funcione correctamente:

```
include "Gramatica";  
include "ObjAnotado";
```

Desde ese punto y en el resto de tu código ya podrás incluir objetos que proporcionen 'anotaciones' a su descripción.

## 3 ¿Qué contiene un Objeto Anotado?

Un objeto anotado no podría ser más simple. Su definición es tal cual la siguiente:

```
Class ObjetoAnotado  
with  
    anotacionNoVacia [; rtrue; ],  
    anotacion 0;
```

Donde los campos significan lo siguiente:

- **anotacion:** debe contener o bien un texto fijo que indica lo que hay que añadir entre paréntesis (o al final de la lista de cosas que la librería pone entre paréntesis) o bien una rutina que la librería debe llamar y que IMPRIMIRÁ el texto correspondiente. Actualmente sólo puede proporcionar UN término, digamos que una sólo frase explicatoria final que agregar a las que pone la librería normal, y debido a limitaciones en el lenguaje y la librería no he encontrado la forma de que esta 'anotación' pudiese ser una lista de anotaciones.
- **anotacionNoVacia:** la librería verifica si 'anotacion' es 0 o no, y en caso de ser 0 no incluye nada tras el objeto, pero en el caso de que 'anotacion' sea una rutina no se puede saber si ésta imprimirá algo o no, por lo que las rutinas estarían obligadas a imprimir siempre algo. Para que esto no sea así la librería al primer llamado a 'anotacionNoVacia', QUE DEBE SER SIEMPRE UNA RUTINA, si esta rutina devuelve true, evaluará 'anotacion', si devuelve 'false' supondrá que 'anotacion' no hubiese escrito nada de haberla llamado.

## 4 Un ejemplo

Veamos un ejemplo con un PNJ, digamos que un Loro que es el animal más usado en las conversacionales. Este loro, que, por cierto, se llama Kwill, es muy pesado y siempre anda revoloteando y espiando nuestras acciones; para incluir esta explicación en el loro podríamos hacerlo simplemente así:

```
ObjetoAnotado Kwill "Kwill, el loro"  
with  
    anotacion "revoloteando sobre tu cabeza para espiar lo que haces";
```

Listo, ahora el loro tendrá la coletilla tras su nombre en cada listado en el que aparezca. Pero claro este comentario no tiene mucho sentido si tienes agarrado al loro. Nada más sencillo de arreglar:

```
ObjetoAnotado Kwill "Kwill, el loro"
with
    anotacion
        [;
            if ( self in jugador )
                "muy triste al verse agarrado por el pescuezo";
            else
                "revoloteando sobre tu cabeza para espiar lo que haces";
        ];
```

Vale ya casi está, (en realidad sólo falta un par de miles de líneas de código para que sea un loro medio creíble ;) ). Pero lo cierto es que no siempre estará el loro tan pesadito, a ratos estará durmiendo, o simplemente no haciendo nada de nada. Esto se puede medir con una variable de estado, por ejemplo numérica. Digamos, por ejemplo que el estado del loro es un número de 0 a 100 (no veas el juego que pueden dar 100 estados diferentes) del 0 al 25 estos estados no tienen mensaje pero el resto sí. Esto se conseguiría mediante:

```
ObjetoAnotado Kwill "Kwill, el loro"
with
    estado elvalorquesea...;
    anotacionNoVacía
        [;
            if ( self.estado > 25 )
                rtrue;
            else
                rfalse;
        ],
    anotacion
        [;
            if ( self in jugador )
                "muy triste al verse agarrado por el pescuezo";
            else
                switch( self.estado )
                {
                    26: "dormido";
                    27: "picoteando comida";
                    ...
                    default: "revoloteando sobre tu cabeza para espiar lo que haces";
                }
        ];
```

Ya sólo tendrás que asociar movimiento y comportamiento a cada uno de esos estados, y crear el grafo de transiciones, etc, etc, etc... O esperarte a que termine mi librería de autómatas y alguien haga un arquetipo de loro mamonazo.

Hasta pronto, espero, ;)