Manual de la librería “Decorados.h”

1 Introducción

La librería de la que trata esta breve documentación fue creada por Zak, insigne miembro de la comunidad aventurera hispana al que le debemos la traducción de Inform al Español, en torno al principio del año 2000. Según creo su intención era exclusivamente usarla en sus propios proyectos, pero tras la publicación de mi “Sonidos.h”, decidimos airear varias librerías que permanecían en el anonimato. Esta es la primera de ellas.

Yo, Mel Hython, particularmente la estoy utilizando masivamente en el desarrollo de mi ANILLO III, dónde me ha resultado particularmente útil.

2 Inicios básicos: ¿Cómo usarla?

Para poder usar la librería de sonidos es necesario añadir un include en tu código de InformatE. Este debe estar detrás del include de la gramática. Así que nos quedaría:

```
include “Gramatica”;
include “Sonidos”;
```

Desde ese punto y en el resto de tu código ya podrás incluir objetos de decorado en tus localidades... como vas a poder ver en las siguientes secciones de este manual.

3 ¿Qué es un decorado? ¿Por qué usarlo?

En casi todas las aventuras de Inform podrás encontrar montones de objetos de *escenario.* Estos objetos son mera ‘decoración’, están ahí para crear atmósfera o por que es lógico que se encuentren ahí, pero no se puede interactuar con ellos excepto para examinarlos.

Cuando se intenta ser ‘estricto’ en cuanto a las cosas que el jugador puede llegar a examinar rápidamente se consigue un árbol de objetos, subobjetos, subsubobjetos... de escenario. Así en una arboleda en un oasis podrás encontrar una palmera, en la que hay un racimo de dátiles, en el que hay un dátil sobre el que se ve una extraña mancha metálica.Todos esos objetos pueden ser meramente ambiente formando así un árbol de objetos de escenario.

Unos ejemplos de jerarquías de objetos de escenario profundo, pero sobre todo de una gran cantidad de objetos de escenario en localidades ricas en ‘cosas’ los puedes encontrar en casi todos los juegos de Inform, en particular puedes usar el fuente de mi juego *Casi Muerto* (http://www.arrakis.es/~meliton/CasiFue.zip).

¿Qué es entonces un decorado? Un decorado no es más que un conjunto de objetos de escenario agrupados bajo un único objeto contenedor. La idea básica es sencilla, en lugar de crear cada uno de los detalles del ‘escenario’ uno a uno como un objeto, los creas todos simultáneamente en un ‘decorado’.

¿Y qué utilidad puede tener esto? Básicamente hay tres razones para usar un objeto de decorado en lugar de una miríada de objetitos con cada detalle del escenario:

1. Por una razón de orden. Es mucho más sencillo crear tan sólo uno o dos objetos de escenario por cada una de las localidades y tenerlos en el código justo al lado de la misma, que crear una larga lista de objetos de escenario.

2. La más importante. El número de objetos en Inform es limitado y su número no demasiado grande (aunque tampoco demasiado pequeño) por lo que cualquier ‘argucia’ que nos permita reducir el número de objetos dentro de la aventura nos permitirán crear aventuras mayores y/o más ricas en ambientación.

3. Utilizando un objeto de escenario es más ‘natural’ crear algunos efectos, como tener un conjunto de elementos de ‘escenario’ presentes en todas las localizaciones del mismo ‘tipo’, como se verá en la sección sexta de este mismo manual.

4 Clase Decorado y su uso

Esta librería crea una clase llamada ‘Decorado’ de la que deben derivar todos los objetos de decorado. Los objetos decorado no deben contener ni ‘nombre_corto’ ni ‘descripcion’, pues usan ambos campos para implementar su función y deben tener una propiedad ‘describir’ que contenga un array de tripletes con una palabra que identifica el elemento del decorado, su descripción y el género del elemento. Es decir, debe tener la forma:

```
Decorado objeto_decorado localizacion_en_la_que_está
with describir
    ‘palabra’ “descripcion” GENERO
    ‘palabra2’ “segunda descripcion” OTRO_GENERO
    ...
    ;
```

Por ejemplo,

```
Decorado decoradoCelda Celda
with describir
    ‘pared’ “Paredes lúgrubes y frías.” G_FEMENINO
    ‘techo’ “Techo lúgrube y alto.” 0
    ‘suelo’ “Suelo lúgrube de piedra.” 0
    ‘paredes’ “Paredes lúgrubes y frías.” G_FEMENINO + G_PLURAL
    ;
```

En este ejemplo se puede observar una de las limitaciones de la librería si hay dos palabras mediante las cuales se podrían referir al mismo objeto es necesario repetir en varias entradas la misma descripción.

El segundo miembro de cada ‘triplete’, es decir, la descripción del elemento del escenario, puede sustituirse por una función **que devuelva una cadena, correspondiente a la descripción a proporcionar**. Esto permite algunos refinamientos como veremos en la sección sexta, pero proporciona una solución sencilla al problema de antes. Si varias palabras pueden servir de ‘nombre’ para un elemento cuya descripción sea extremadamente larga se puede crear una función muy simple que sólo devuelva la cadena con la descripción:

```
[ DescripcionPared;
    return
        “Antaño estas debieron de ser regias paredes de un castillo,
        un baluarte de fortaleza y resistencia que intimidasen a los
        enemigos. Ahora sólo son viejas ruinas.”;
];
```

Con lo que bastaría poner un decorado en la forma:

```
Decorado decoradoRuinas LugarDeLasRuinas
with describir
    ‘pared’ DescripcionPared G_FEMENINO
    ‘paredes’ DescripcionPared G_FEMENINO + G_PLURAL
    ;
```

5 Clases de Decorado ‘mejoradas’

Veamos algunas cosas adicionales que se pueden hacer con los decorados. Lo primero que tienes que saber es que en los objetos de decorados se puede implementar el método ‘antes’. En la clase básica que contiene la librería se permite al jugador Examinar cualquiera de los componentes del decorado, pero cualquier otra acción provoca el mensaje ‘Déjalo, sólo es decorado.’. Que no está muy mal, pero personalmente encuentro muy frustante porque al indicarme tan claramente que es un objeto de decorado rompe un poco la ‘magia’.

Por suerte puedes cambiar el comportamiento del decorado de una forma extremadamente sencilla. Basta conque crees una clase de decorado hija de la básica, redefinas las acciones que quieras (deja a rfalse la de Examinar) e instancia tus objetos de decorado de esta nueva clase en lugar de usar la básica. Así por ejemplo en el ANILLO III, existe una clase DecoradoAmpliado en la forma:

```
Class DecoradoAmpliado
class Decorado
with
antes [;
  Examinar: rfalse;
  Coger: "No creo que tenga mucho sentido cargar con esa cosa inútil.";
  Empujar: "Esa cosa no tiene mucha pinta de poder ser empujada.";
  Oler: "No parece que huela a nada especial.";
  Escuchar: "No produce ningún sonido.";
  BuscarEn: "No hay nada que buscar en eso.";
  default: "No veo ninguna razón para hacer éso.";
  ];
```

En realidad se puede hacer mucho mejor. En el atributo ‘cantidad’ el decorado dispone de la palabra que ha activado el acceso a la clase decorado, es decir, a la palabra a la que se está refiriendo el jugador. Esto se puede utilizar para afinar aún más los mensajes, así por ejemplo se podría hacer un decorado en la forma:

```
Decorado decoradoCelda Celda
with
    describir
        ‘pared’ “Paredes lúgubres.” G_FEMENINO
        ‘techo’ “Techo lúgubre.” 0,
    antes
        [;
        Empujar:
        if ( self.cantidad == ‘techo’ ) “No llegas.”
        else “Deja “, (el)self, “ en paz.”;
        ];
```

6 Usos ‘avanzados’ de los decorados

Con lo que se ha visto hasta el momento es posible sacarle un gran partido a los objetos de la clase decorado, pero se pueden conseguir algunos efectos adicionales:

6.1 Decorados asociados a la clase de la localización

Esta es posiblemente la más interesantes de todas las aplicaciones ‘avanzadas’. Supongamos que existe en un conjunto de localizaciones en tu aventura que pertenecen al mismo grupo o zona. Todas ellas son parte de las catacumbas o bien de un pantano o de un bosque, etc... Lo normal sería que en todas ellas haya un conjunto de elementos omnipresentes que el jugador puede querer Examinar en cada una de ellas. Normalmente esto exigiría una aburrida larga lista de objetos de escenario, pero con la clase de Decorado y el uso del atributo ‘esta_en’ bastaría con algo como esto (extraído del ANILLO III):

```
Class Pantano
    class Lugar;


DecoradoAmpliado dec_EnPantano
with
    esta_en
    [;
    if ( localizacion ofclass Pantano )
        return true;
    else
        return false;
    ],
    describir
        ’turba’ "No es más que materia vegetal en profunda descomposición." 0
        ’pantano’ "Esto que te rodea, húmedo, putrefacto, turgente de vida y de muerte... en una palabra cómodo como un lecho de monedas, es el pantano." 0
        ’fango’ "El pantano no sería tal si no hubiese enormes cantidades de fango y turba por todas partes." 0
        ’charcas’ "No es más que agua sucia a veces muy profundas." G_FEMENINO+G_PLURAL
        ’pozos’ "Parece suelo, pero es una trampa mortal." G_PLURAL
        ’arenas’ "Parece suelo, pero es una trampa mortal." G_FEMENINO+G_PLURAL
        ’nubes’ "Prefieres no mirarlas mucho, te hacen recordar el tiempo en el que volabas libremente entre ellas." G_FEMENINO+G_PLURAL;
```

Es decir, de un plumazo hemos conseguido que todas y cada una de nuestras localidades de la clase ‘Pantano’ dispongan de los elementos ‘turba’, ‘pantano’, ‘fango’, ‘charcas’, ‘pozos’, ‘arenas’ y ‘nubes’ para que el jugador pueda examinarlos. Evidentemente esta implementación sólo proporciona estos elementos en las localidades en las que se encuentra el jugador (debido al uso de ‘esta_en’), pero a fin de cuentas normalmente los decorados sólo se quieren para el jugador.

Si programas un PNJ, que deambula por ahí y va examinando escenarios, por favor ¡no te olvides de mandarme una copia de la aventura!, ya sabes, meliton@arrakis.es :-)

6.2 Decorados ‘flexibles’

Este es un uso casi ‘no avanzado’. Como ya se ha dicho la descripción de los objetos del decorado se puede redefinir por una función, esto permite que la descripción pueda variar según el estado de estos objetos.

Por ejemplo, digamos que entre el decorado de la habitación hay una pizarra sobre la que se puede Dibujar un conjunto de figuras geométricas. Supongamos que estas figuras se definen de la clase Figura y que son objetos normales y sólo puede haber una cada vez. Se podría declarar un decorado en la forma:

```
Decorado decoradoClase Clase
with
figuraDibujada 0,
describir
    ‘mesas’ “Las mesas que recuerdas de tus tiempos de colegio.” G_FEMENINA + G_PLURAL
    ‘pizarra’
    [;
    if ( self.figuraDibujada == 0 )
        return “Limpia, de un profundo color verde.”;
    else if ( self.figuraDibujada ofclass Figura )
        return self.figuraDibujada.vistaEnPizarra();
    ] G_FEMENINA,
antes [;
    Dibujar:
    ...
```

6.3 Decorados ‘activos’

Lo que voy a explicar a continuación es ‘extremadamente’ inapropiado y poco elegante, pero a veces si el número de objetos es muy elevado o por algún motivo especial puede estar justificado.

Como ya he dicho se puede redefinir el comportamiento de las acciones y además se puede distinguir la palabra a la que se está refiriendo el jugador en cada momento, esto se puede usar para implementar verdaderas ‘acciones’ sobre los objetos del decorado.

Veamos un ejemplo:

```
Decorado decoradoCelda Celda
with
contadorExaminas 0,
describir
    ‘suelo’ 0 0
    ‘pared’ “Lúgubre.” G_FEMENINO,
antes
    [;
    Examinar:
    if ( self.cantidad ~= ‘suelo’ )
        rfalse;
    self.contadorExaminas++;
```

```
if ( self.contadorExaminas == 1 )
    “Está hecho de piedras. Ninguna parece suelta, pero hay tantas.”;
else if ( self.contadorExaminas == 2)
    “No parece haber piedras sueltas pero aún faltan muchas por
    mirar...”;
else if ( self.contadorExaminas == 3)
{
    move PiedraSuelta to localizacion;
    “¡Eureka! ¡Hay una suelta en una esquina! La paciencia siempre es
    recompensada.”;
}
else
    “No, no parece que haya más piedras sueltas.”;
];
```

Como se puede ver, en este escenario se exige que el jugador examine repetidamente el suelo hasta encontrar una piedra suelta. O sea, que se pueden crear comportamientos bastante complejos a base exclusivamente de objetos de decorado. Pero por claridad de código recomiendo encarecidamente que se utilice esta clase de trucos lo menos posible, para proporcionar algo tan complejo como esto es preferible crear un objeto propio para el ‘Suelo’.

Y nada más, que disfrutéis de la librería.