Superglús - txtPAWS

Programa: Baltasar

Manuales : Uto (Basándose en el manual de txtPAWS)

Nota importante: los contenidos del presente manual hacen referencia a conceptos contenidos en el manual Introducción y la Gúia Técnica de Superglús. La falta de comprensión de determinados conceptos puede dar como resultado la no comprensión de este manual, por lo que se sugiere la lectura previa de dichos manuales.

Introducción

En los viejos tiempos de los ordenadores de 8 bits, época de nacimiento de los parsers del tipo de Superglús (PAW, QUILL), el programador se veía obligado a recordar con demasiada frecuencia determinados números (números de flag, números de objeto, números de mensaje, etc.). En la práctica los programadores de dichos sistemas mantenían a su lado, escrito sobre papel, dicha información.

Con la llegada de los PCs y los sistemas de 16 y 32 bits al menos el parser NMP (y quizá otros como SKC y SINTAC, pero no puedo asegurarlo) trataron de evitar dicha lista permitiendo utilizar lo que llamaríamos 'etiquetas'. Dichas etiquetas consistían en una serie de letras (habitualmente formando una palabra o acrónimo) que, al ser encontrados , eran substituidos por un número previamente definido.

Así por ejemplo en NMP, si en la sección denominada 'constantes' se definia:

LINTERNA=0

Posteriormente en el código podía perfectamente ponerse "CARRIED LINTERNA" en lugar de "CARRIED 0", porque para el parser era lo mismo.

Evidentemente la capacidad de referirnos a los objetos, localidades, mensajes, por un nombre, facilitaba sobre manera la programación, al no tener que recordar decenas de números.

Superglús sin embargo no incluye esa posibilidad, soportando sólo números, pero he aquí que txtPAWs viene en ayuda de Superglús para dotarle de dicha posibilidad.

¿Qué es txtPAWs?

txtPAWs es lo que se denomina un preprocesador: toma un fichero que contiene un código en un determinado lenguaje de programación y da como salida un texto en otro lenguaje parecido.

En nuestro caso txtPAWs tomará un fichero con extensión TXP en el que podremos ver cosas como:

CARRIED &&linterna

Y dará como salida un fichero con extensión SCE para Superglús con cosas como:

CARRIED 0

Definiendo etiquetas

La definición de etiquetas indicará que textos son substituidos por qué números en concreto, y en general dichas definiciones deben ser colocada dentro del fichero SCE antes de la sección de control /CTL (o en la solapa 'Definiciones' si usamos el editor de Superglús)

Pongamos un ejemplo:

##define obj linterna 0

Esto hace que 'linterna' sea equivalente a '0' allí donde aparezca, y además se le define como una relación relativa a un objeto (por lo de obj).

La lista completa de tipos de relación existentes es la siguiente:

Obj	Objeto
Flg	Flag, bandera
Loc	Localidad
Msg	Mensaje de usuario
Snd	Efecto de sonido
Msc	Musica ambiental ligada a localidad
Grf	Gráfico
Pic	Gráfico ligado a localidad
Const	General, no va ligado a nada y se puede usar en cualquier sitio (como objeto, flag, valor, etc.)
Macro	Macrosubstitucion: en lugar de un numero va una cadena de caracteres, es decir, un texto cualquiera, que será substituido cuando aparezca la etiqueta.

Usando etiquetas

Para usar una etiqueta basta con poner el texto que hacia referencia precedido de dos caracteres “&”. Es decir:

Si hicimos

##define flg numero_monedas 100

Luego lo usamos así:

```
CONTAR MONEDAS    WRITE “Tienes “
                        PRINT &&numero_monedas
                        WRITE “ monedas.”
                        DONE
```

Uso detallado

Se describe ahora un ejemplo para cada uno de los tipos, para más claridad. Tambien se indica en que sección se define adicionalmente en aquellos casos que pueda no ir al principio (seccion declaraciones en el editor):

- obj = objeto
  Se define en la sección de objetos. Permite referirse a un objeto (un número de 0 a 255) mediante un identificador.

  ```
  /OBJ
  /0 1 1 _ _ candelabro _
  ##define obj candelabro 0
  ```

  Más tarde, puede reemplazarse con:

  ```
  ENCIENDE CANDELABRO PRESENT &&candelabro
              MESSAGE &&candelabro_encendido_desc
              DONE
  ```

- flg = bandera
  Se define en la sección de procesos (es el único tipo que se permite definir en la sección de procesos). Permite referirse a un bandera/flag (un número de 0 a 255) mediante un identificador.

  ```
  /PRO 0
  ##define flg puerta_abierta_sw 80
  ```

  Más tarde, puede reemplazarse con:

  ```
  ABRE PUERTA AT &&biblio
          PRESENT &&llave
          EQ &&puerta_abierta_sw 1
          MESSAGE 100
  DONE
  ```

- loc = localidad
  Se define en la sección de localidades. Permite referirse a una localidad (un número de 0 a 255) mediante un identificador.

  ```
  /LOC
  /1
  ##define loc biblio 1
  La biblioteca de esta gran casa.
  ```

  Más tarde, puede reemplazarse con:

  ```
  ABRE PUERTA AT &&biblio
          PRESENT &&llave
          EQ &&puerta_abierta_sw 1
          MESSAGE &&puerta_abierta_desc
          DONE
  ```

- msg = mensaje
  Se define en la sección de mensajes. Permite referirse a un mensaje (un número de 0 a 2^31) mediante un identificador.

```
/MSG
/1
##define msg puerta_abierta_desc 1
Has abierto la puerta
```

Más tarde, puede reemplazarse con:

```
ABRE PUERTA AT &&biblio
PRESENT &&llave
EQ &&puerta_abierta_sw 1
MESSAGE &&puerta_abierta_desc
DONE
```

- snd = sonido (se reproduce mediante el condacto *BEEP*)
  Se define en la sección denominada DEF (ya que no está asociado ni a una localidad ni a ninguna otra entidad).

  ```
  ##define snd puerta_abriendose.mod 0
  /CTL
  ```

  Más tarde, puede reemplazarse con:

  ```
  ABRE PUERTA AT &&biblio
  PRESENT &&llave
  EQ &&puerta_abierta_sw 1
  MESSAGE &&puerta_abierta_desc
  BEEP &&puerta_abriendose.mod 0
  DONE
  ```

- msc = música ambiental (se reproduce automáticamente al entrar en la localidad)
  Se define en la sección de localidades. Permite referirse a un fichero de música asociado a una localidad (un número de 0 a 255) mediante un identificador.

  ```
  /LOC
  /1
  ##define loc biblio 1
  ##define msc biblio.aif 1
  La biblioteca de esta gran casa.
  ```

  La música será reproducida al entrar en la habitación "biblio". Nótese que el identificador **debe** coincidir con el nombre del fichero.

- pic = gráfico (se pinta automáticamente al entrar en la localidad) Se define en la sección de localidades. Permite referirse a un gráfico para una localidad (un número de 0 a 255) mediante un identificador.

  ```
  /LOC
  /1
  ##define loc biblio 1
  ##define msc biblio.aif 1
  ##define pic biblio.jpg 1
  La biblioteca de esta gran casa.
  ```

  El gráfico será pintado al entrar en la habitación "biblio". Nótese que el identificador **debe** coincidir con el nombre del fichero.

- grf = gráfico (no está relacionado con ninguna localidad)
  Se define en la sección denominada DEF (ya que no está asociado ni a una localidad ni a ninguna otra entidad).
  La zona DEF es la zona situada entre el comienzo del fichero y la línea /CTL. Es decir, el

comienzo del fichero. Permite referirse a un gráfico (un número de 0 a 2^31) mediante un identificador.

```
##define grf puerta_abierta.jpg 1
/CTL
```

Más tarde, puede reemplazarse con:

```
ABRE PUERTA AT &&biblio
PRESENT &&llave
EQ &&puerta_abierta_sw 1
MESSAGE &&puerta_abierta_desc
PICTURE &&puerta_abierta.jpg
BEEP &&puerta_abriendose_snd 0
ANYKEY
DESC
DONE
```

Nota: el código 0 se refiere siempre al gráfico que se pinta cuando el jugador está a oscuras (de existir).

- const = constante (no guarda relación con objetos, localidades ...)
  Se define en la sección denominada DEF (ya que no está asociado ni a una localidad ni a ninguna otra entidad), o bien en cualquier sección /PRO.

  ```
  /PRO
  ##define const NUM_MAX_OBJETOS 5
  ```

  Más tarde, puede reemplazarse con:

  ```
  COGE _ LT &&num_objetos_en_jugador &&NUM_MAX_OBJETOS
      AUTOG
      DONE
  ```

- macro = macrosustitución (no guarda relación con objetos, localidades ...)
  Se define en la sección denominada DEF (ya que no está asociado ni a una localidad ni a ninguna otra entidad), o bien en cualquier sección /PRO.

  ```
  /PRO 0
  ##define macro if(jugador_en_biblioteca) "AT &&biblio"
  ```

  Más tarde, puede reemplazarse con:

  ```
  COGE CERILLAS &&if(jugador_en_biblioteca)
              MESSAGE &&coges_las_cerillas_desc
              AUTOG
          DONE
  ```

  NOTA: La macrosustitución debe ocupar una sola línea.

Include

Adicionalmente txtPAWs permite que dentro de un fichero fuente cualquiera se incluya otro. Lo mostraremos con un ejemplo para más claridad:

Si en la tabla de respuestas aparece, en cualquier sitio:

##include respuestas_comunes.txp

es como si el contenido del fichero respuestas_comunes.txp estuviera copiado en el lugar del incluye.

Esto permite, en el caso de estar usando Superglús desde la línea de comandos, trabajar con varios ficheros, cada uno con una seccion, o separarlo como nosotros veamos. Lo cierto es que con el editor de Superglús esto no tiene demasiado sentido, así que no nos extenderemos más en este punto.

Graficos y sonidos en Superglús

Una aventura generada por Superglús, o por cualquier otro sistema cuyo destino sea la máquina Glulx, puede presentarse en dos formatos:

ULX: Consistente en la aventura compilada lista para ejecutarse.

BLB: Consistente en la aventura compilada lista para ejecutarse, más los recursos gráficos y de sonido necesarios para la misma.

Para obtener un fichero BLB necesitamos el compilador de ficheros blorb, mas conocido como BLC, e indicarle que ficheros queremos que se metan en el fichero BLB.

Afortunadamente txtPAWs también es de gran ayuda para esto, dado que él mismo, según se van definiendo graficos y musicas va creando el fichero que le dice a BLC como hacer el fichero BLB.

Por ejemplo:

##define msc biblio.aif 1

Además de definir la musica para la localidad 1, pone el biblio.aif en dicho fichero, que siempre será de igual nombre que el fichero .TXP de entrada pero con extension.blc.

Al final de la compilación, tras pasar txtPAWs, SCE y Glulxa, si ejecutamos blc tendremos también el fichero .blb adecuado.

Afortunadamente el editor de Superglús se encarga de todo eso y no tendrémos que pensar en como usarlo a no ser que no usemos el editor.