The Professional Adventure Writing System

Introduction

CPM 2.2
or
CPM+/3

Reconocimiento

Gracias a Jill por aguantar las muchas horas que he pasado machacando el teclado.

Contenidos

Introducción

Bienvenido al mundo de la creación de aventuras...

El Professional Adventure Writer (o PAW como se le conoce más comúnmente) te proporciona los medios para crear aventuras de alta calidad (en código máquina) o de igual o mejor calidad que muchas comerciales.

PAW te proporciona la estructura básica para crear un juego, pero también te proporcionará una historia imaginativa y originales puzzles.

Los manuales suministrados con PAW cubren todos los aspectos de su uso. Este manual proporciona un tutorial sobre la creación de una aventura y te recomendamos que lo sigas, así como los ejemplos que lo acompañan antes de intentar crear tu propia aventura. El otro manual te proporciona una vista detallada de todo el sistema y puede ser usado como guía de referencia mientras escribes tus propios juegos.

Buena suerte...

Gran cantidad de tiempo y esfuerzo ha sido puesto para asegurar que PAW funcione con todas las situaciones concebibles de una forma lógica y útil. El resultado es un programa de unos 60 Kb de tamaño, y entra dentro de los posible que en alguna parte en lo más profundo del código se hayan quedado algunos errores bien escondidos. De hecho, un conocido dicho dice: “El testeo sólo prueba la existencia de errores, no su ausencia”. Si encontraras algún problema, por favor, infórmanos para que lo solucionemos si es necesario.

Todo el cuidado necesario ha sido puesto en la preparación de estos manuales y los programas que los acompañan. Sin embargo, los autores y Gilsoft International Ltd no asumen responsabilidad alguna por errores, omisiones o aplicabilidad de sus contenidos para aplicación alguna. Tampoco asumimos posibles daños derivados de su uso. Esta renuncia no afecta a nuestros derechos estatutarios.

Comenzando

La versión CP/M de PAW consta de un Compilador y un Intérprete. Las aventuras se crean introduciendo toda la información del juego en un archivo de texto conocido como archivo fuente y usando entonces el Compilador para generar la base de datos. El Intérprete entonces utiliza la información en la base de datos para proporcionar la aventura terminada.

Se requieren algunos conocimientos sobre CP/M:

a) Debes entender el formato de los nombre de archivo de CP/M, por ejemplo d:nnnnnnnn.ttt.
b) Debes saber mostrar el contenido de un directorio con tus archivos, por ejemplo DIR.
c) Debes saber borrar un archivo, por ejemplo ERA ANTIGUO.TXT.
d) Debes saber renombrar un archivo, por ejemplo REN NUEVO.SCE=ANTIGUO.SCE
e) Debes saber copiar un disco completo para obtener una copia de seguridad.
f) Debes saber duplicar un archivo, por ejemplo, para hacer una copia de START.SCE que se llame TICKET.SCE debes usar PIP TICKET.SCE=START.SCE.
g) Si tienes más de una unidad de disco debes saber referirte a un archivo en una unidad determinada y copiar archivos de una una unidad a otra, por ejemplo PIP B:A:TICKET.SCE.

Antes de ir más allá asegúrate de que has hecho una copia de seguridad del disco de PAW y que has puesto el original a buen recaudo.

Es conveniente leer el archivo READ.ME del disco en este momento para ver los añadidos y/o correcciones que se hayan hecho a este manual desde que fue impreso. Esto se puede conseguir con el comando CP/M TYPE:

```
TYPE READ.ME
```

Instalando el Intérprete.

Antes de que puedas usar el Intérprete, éste necesita ser instalado. El Intérprete necesita saber, entre otras cosas, el número de columnas de tu pantalla, el número de líneas de tu pantalla y cómo limpiarla. Para instalar el Intérprete escribe: PAWINST PAWINT.COM

Tu pantalla debe parecerse a algo como ésto:

```
Columns       is set to 90
Lines         is set to 31
Timeout       is set to 4680
Clear screen  is set to '1B45148' (hex)
Pause         is set to 17
```

Enter A to change Columns
Enter B to change Lines
Enter C to change Timeout
Enter D to change Clear screen
Enter E to change Pause
or Enter F to Exit
Choose A-F?

No te preocupes por los valores de Timeout y Pause por el momento ya que estos son para 'ajuste fino'. Comprueba que los valores para Columns, Lines y Clear screen son los correctos para tu ordenador y si no, corrígelos. Nota que el código para Clear screen ha de ser introducido en hexadecimal y que los códigos introducidos deberían limpiar la pantalla y colocar el cursor en la esquina superior izquierda de la pantalla.

Algunos ejemplos de valores son:

Ordenador	BBC con Z80 (2º procesador)	PCW8512 CP/M 3	CPC464 CP/M 2	CPC6128 CP/M +
Modo pantalla	MODE 3	24x80 off	MODE 2	MODE 2
Columns	80	90	80	80
Lines	25	31	25	24
Timeout	5461	4680	1705	1705
Clear screen	'0C'	'1B451B48'	'0C'	'1B451B48'
Pause	35	17	17	17

Cuando estés satisfecho con los valores correctos para tu ordenador, teclea 'F' para salir y se te preguntará “Do you want to update the file on disc?”.Responde Y si has hecho algún cambio.

Elige tu Editor de Texto

Para crear tu aventura necesitarás un Editor o Procesador de Texto que usará para producir archivos 'No-Documento', por ejemplo archivos de texto de CP/M sin códigos de control. ED80 de Hisoft, Protext o Worstar en modo no-documento son posibles opciones. Si ya tienes un editor de texto apropiado, úsalo. Si no, necesitarás usar el que que proporcionamos, que es muy básico. Si eliges usarlo, las instrucciones para su instalación y uso están en el Apéndice A de este manual, necesitarás familiarizarte con su uso antes de comenzar.

Nota: Cuidado con algunos procesadores de texto que utilizan símbolos inusuales al inicio de las líneas como caracteres de control, por ejemplo, los comandos 'DOT' de Wordstar y el '>' de Protext. Este último causará problemas con uno de los mensajes de sistema (33) que es prompt usado para la entrada del jugador. Pon un espacio delante del '>' o utiliza un otro carácter para evitar conflictos.

Un disco de trabajo

Para obtener el máximo rendimiento del sistema PAW, te sugerimos que crees un disco de trabajo que contenga sólo aquellos archivos que sean absolutamente necesarios, y algunas aplicaciones de

CP/M útiles. Esto es especialmente importante en sistemas CP/M 2 de una sola unidad, donde el uso de múltiples discos es prácticamente imposible. Nuestras sugerencias para varios sistemas son las siguientes:

CP/M 2.2 de una unidad (por ejemplo Amstrad 464 con unidad de disco o Amstrad 664).

Formatea un disco con la opción system, y así puedes usar CTRL C para reiniciar el sistema de disco. Copia entonces STAT en el disco. Si estas usando tu propio editor de texto, cópialo también, y si no copia los instalados EDIT.COM y EDIT.INS del disco de PAW. Finalmente copia los siguientes archivos del disco de PAW:

PAWCOMP.COM
PAWINT.COM (después de instalarlo correctamente)
START.SCE

Cuando comiences un juego simplemente renombra START.SCE con el nombre del juego. Esto te proporcionará la máxima cantidad de espacio para tus archivos Fuente y Base de Datos. STAT puede ser dejado fuera si no quieres saber el espacio libre del disco.

CP/M 2.2 o CP/M 3 con dos unidades (por ejemplo Acorn Z80, PCW 8512).

Aquí no hay problema. Crear un disco de sistema con cualquier útil aplicación de CP/M y copia los archivos principales de PAW en él. Es una buena idea dejar unos 50 Kb libres en el disco para recibir tus archivos de Base de Datos. Dejas así la segunda unidad totalmente libre para el Fuente. Notarás al COMPILADOR más rápido si compilas la Base de Datos a una unidad diferente a la que contiene el Fuente.

Sistemas CP/M 3 de una unidad (por ejemplo CPC 6128)

Con las facilidades de disco virtual de CP/M 3 puedes almacenar todos tus archivos Fuente y Base de Datos en un disco aparte y referirte a él como unidad B. No intentes compilar la Base de Datos en una unidad virtual diferente a la del Fuente o puedes acabar con un brazo dormido de tanto cambiar discos.

Sistemas con RAMdisc (por ejemplo la series 8000 y 9000 de PCW)

Crea un disco de trabajo de sistema como con los sistemas CP/M 2.2 de una unidad, pero con PIP también. Entonces, al inicio de una sesión simplemente copia los contenidos de este disco en la RAM (por ejemplo, en un PCW harías PIP M:=A:*.*) y cambia entonces el disco de la unidad A: por tu disco Fuente. Haz que la unidad RAM sea la unidad actual y listo. No te sientas tentado de usar la unidad RAM para contener tu Fuente, a pesar de su velocidad es volátil y el CEGB no da garantías.

Conceptos

Probablemente sea una buena idea introducir en este punto algunos de los más importantes conceptos que PAW encarna en su diseño.

El Archivo Fuente

Es una colección de tablas que contienen toda la información, en un formato legible por humanos, para definir la aventura. El archivo START.SCE es un ejemplo de un archivo Fuente y es un útil punto de partida para una aventura.

El Compilador

PAWCOMP.COM es un programa que comprueba el archivo Fuente en busca de errores y lo convierte en una Base de Datos PAW.

La Base de Datos

Ésta contiene la salida del Compilador y es una versión legible por la máquina del archivo Fuente. Puede por tanto considerarse una aventura compilada. START.PDB es un ejemplo de archivo de Base de Datos.

El Intérprete

PAWINT.COM es la parte principal del PAW. Contiene todas las rutinas en tiempo de ejecución necesarias para una aventura terminada. Esencialmente captura las órdenes del jugador y utiliza la información de la Base de Datos para decodificarla y responder a esas órdenes.

Parser

Vuelta a la escuela para ésto:

parse v.t Clasificar una palabra o analizar una frase en términos gramaticales. (Diccionario de Inglés Minster)

PAW viene con un potente Parser para convertir lo que el jugador teclea cuando juega tu aventura en una serie de 'Frases Lógicas' simplificadas para las cuales habrás definido las respuestas. El Parser hace ésto extrayendo 'frases' de la cadena introducida una por una y permitiendo al resto de PAW interpretar su significado. Las frases se separan por cualquier signo de puntuación y las conjunciones 'AND' o 'THEN' – aunque puedes cambiar esto si es necesario. Cuando se acaban las frases en la cadena introducida actual pedirá otra. Una frase consta al menos de un Verbo (palabra que indica acción) y opcionalmente dos Nombres (palabras que nombran objetos) posiblemente con Adjetivos asociados (que describen los objetos), un Adverbio (que modifica el verbo), una preposición (que muestra la relación de un Nombre con otra palabra) y finalmente una cadena entre comillas que se usa para dirigirse a otros personajes en la aventura.

Escribiendo una aventura

Ahora, comienza la diversión...

Planificación

Planificar tu juego es muy importante si quieres conseguir un resultado profesional. De nada sirve sentarse ante la máquina y escribir ajustes y comienzos mientras esperas la inspiración. Seguramente te enredarás en un laberinto de números y letras sin recursos excepto empezar de cero en cualquier caso.

Para ilustrar el enfoque recomendado para escribir una aventura veremos el diseño y desarrollo de un juego sencillo desde la idea inicial hasta las pruebas finales.

Conseguir una idea

Esto es siempre la parte más dura en la creación de cualquier cosa. Un argumento original puede dar lugar a un juego con un interés que rescatar a una princesa quizá no consiga en un aventurero moderno.

Hay temática para aventuras a nuestro alrededor en muchas acciones cotidianas, en lugares exóticos del mundo y de fuera de este mundo.

Si decides basar un juego en un libro o película que disfrutaste y quieres darle uso comercial, asegúrate de obtener permiso del autor original o de los propietarios del copyright.

Para nuestro ejemplo usaré un problema sencillo que ubica a un pasajero de camino a casa:

Mientras espera en la parada del autobús, el ticket del pasajero es arrastrado por la brisa y un pajarito lo lleva a un parque cercano. El ordenador reproducirá la parte del pasajero el cuál debes dirigir para encontrar el ticket antes de que llegue el autobús.

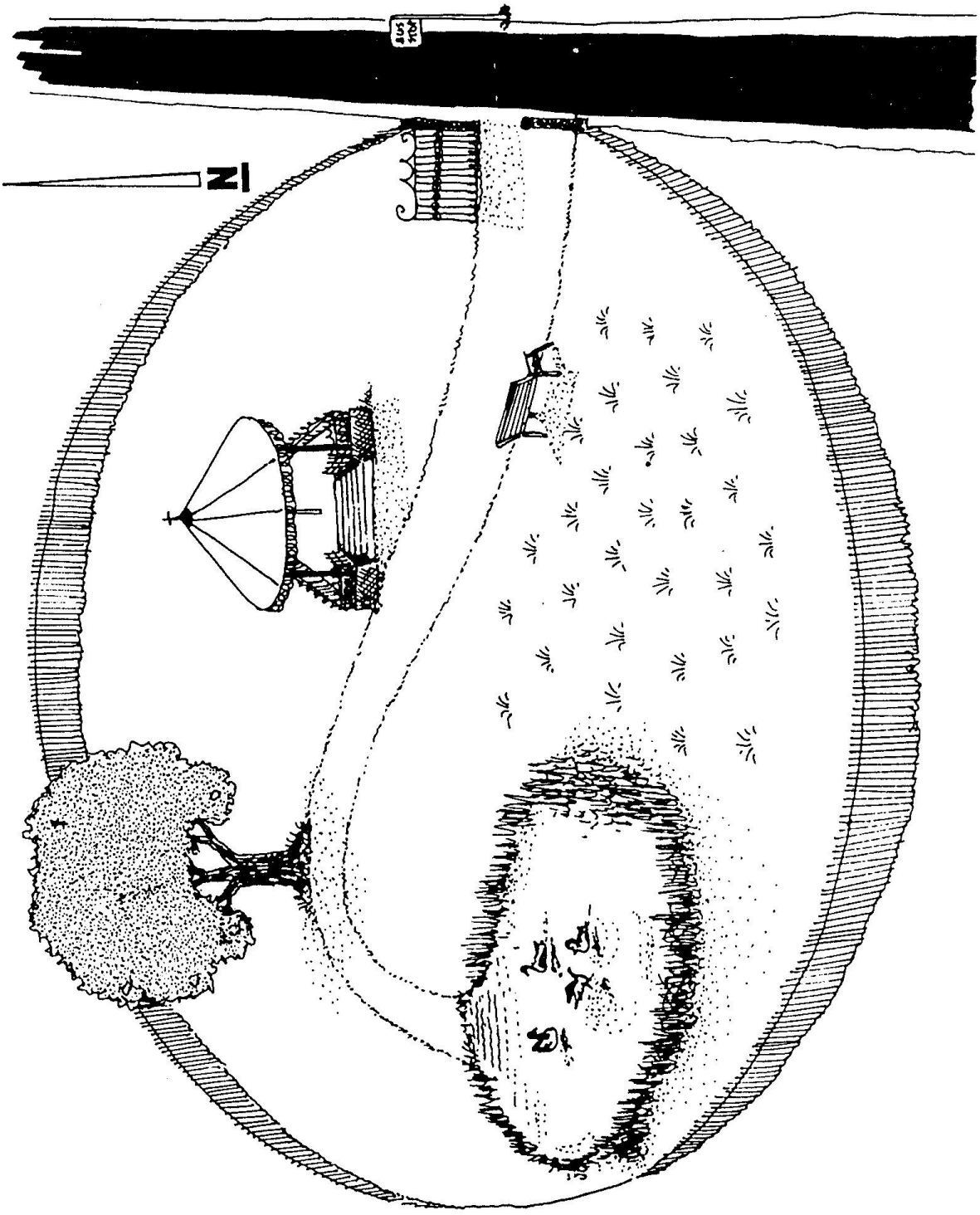
Diseño del juego

Ahora que tienes la idea, merece la pena dibujar un boceto de la zona del juego en la que el juego tiene lugar, como hemos hecho en la figura 1 (bueno en realidad lo hizo nuestro artista...)

Fíjate que cualquier diseño de un juego debería estar dentro de un zona lógicamente acotada porque si no el jugador se preguntará por qué no puede ir en una dirección cuando nada parece bloquear el paso.

Una aventura consta de un número de ubicaciones discretas, esto es, independientes, o sitios que el jugador puede visitar. Ahora debes decidir qué zonas pueden ser una ubicación y numéralas individualmente.

Figura 1



Escribiendo una aventura

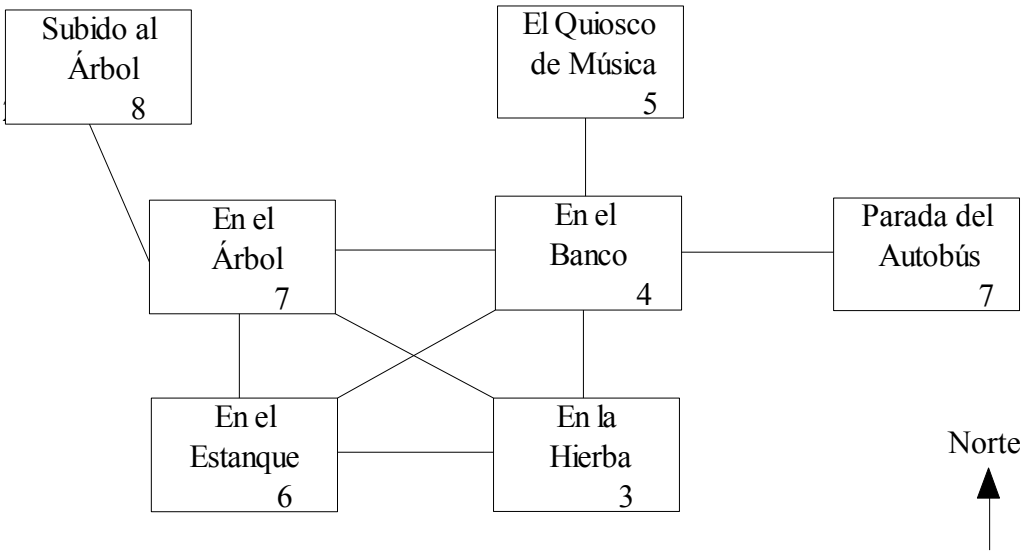
Intenta hacer la escala consistente o lógica (si no el juego será ilógico), ya que un simple paso para llegar a un aeropuerto anteriormente descrito como a 10 millas no ayuda a dar sensación de realismo. Puedes introducir un medio de transporte como un taxi por ejemplo, si fuera necesario. Ahora la ubicación 0 está siempre reservada como una pantalla de título de un juego, y queremos la ubicación 1 para un uso especial posterior, así que numeraremos las ubicaciones del 2 en adelante.

Para nuestro ejemplo hemos elegido las 7 ubicaciones siguientes:

2 La Parada del Autobús
3 En la Hierba
4 En el Banco
5 El Quiosco de Música
6 El Estanque Ornamental
7 Bajo el Árbol
8 Subido al Árbol

Para clarificar el diseño y calcular los posibles movimientos, la figura 2 es un diagrama de bloques mostrando un estilizado mapa del juego.

Figura 2



Ahora comenzaremos a escribir la descripción textual de cada ubicación. Intenta no hacer de ellas un seco y nada interesante monólogo del estado de la nación. La concisión y brevedad son igual de efectivas para crear ambiente si no ejercitas mucho la imaginación. Recuerda utilizar una forma para dirigirte ('Yo' o 'Tú', normalmente) o el jugador sufrirá una seria crisis de identidad. Nota que si utilizas 'Tú' como forma de dirigirte, necesitarás cambiar el sistema de mensajes – mira la guía técnica para detalles.

Comenzando a Escribir

Vamos a llamar a esta aventura TUTORIAL por lo que necesitaremos crear un archivo Fuente llamado TUTORIAL.SCE. Como punto de partida haz una copia de START.SCE llamada TUTORIAL.SCE. Esto puede conseguirse usando el programa PIP bajo CP/M. Por ejemplo PIP TUTORIAL.SCE=START.SCE. Ahora edita TUTORIAL.SCE con tu propio editor o con el que nosotros proporcionamos. En el caso de EDIT, debes escribir EDIT TUTORIAL.SCE.

El archivo Fuente consta de un número de secciones que describen la aventura. Vamos a echar un vistazo rápido al archivo Fuente para ver que hay allí. La primera línea es un comentario, que nos dice que este es un archivo Fuente START y la fecha en la que se editó. Cualquier línea que comience por punto y coma es un comentario. Los comentarios son totalmente ignorados por el Compilador y por tanto no ocuparán sitio en el archivo de Base de Datos cuando se genere. Poner muchos comentarios te ayudará a recordar que va sucediendo en tu juego.

La sección CONTROL comienza con una línea que empieza con '/CTL' y da alguna información de control para el Compilador.

La sección VOCABULARIO comienza con una línea que empieza con '/VOC' y define todas las palabras que PAW 'entenderá', por ejemplo TERMINAR.

La sección TEXTO DE MENSAJES comienza con una línea que empieza con '/MTX' y define los mensajes realmente necesarios para la aventura, por ejemplo “Es un sandwich de queso !!”

La sección TEXTO DE UBICACIONES comienza con una línea que empieza con '/LTX' y define las descripciones que se usarán para cada ubicación en la aventura, por ejemplo “Estás en una celda. No hay manera de salir.”

La sección CONEXIONES comienza con una línea que empieza con '/CON' y define las conexiones entre todas las ubicaciones de la aventura.

La sección DEFINICIÓN DE OBJETOS comienza con una línea que empieza con '/OBJ' y define cada objeto de la aventura, por ejemplo Objeto 0 empieza la aventura siendo llevado y tiene un peso de 1 unidad, no es un contenedor y no puede llevarse puesto, etc

La sección TABLA DE PROCESO comienza con una línea que empieza con '/PRO 0' y proporciona el control principal del juego, por ejemplo cuando el jugador escribe MIRAR, la acción DESC es

llevada a cabo para describir la ubicación actual.

Volviendo con nuestro tutorial, primero cambia el comentario de la línea 1 para que diga:

;TUTORIAL archivo Fuente DD/MM/YY

Donde DD/MM/YY debe ser reemplazado con el Día, Mes y Año actuales. Es una buena idea indicar las versiones de archivo de esta manera, así puedes saber siempre si estás trabajando con la última versión.

Mientras estamos en la parte superior del archivo, la línea 3 dice al Compilador que escriba la Base de Datos compilada en la unidad A. Puedes cambiar esto si prefieres otra unidad.

Ahora vamos a concentrarnos en las ubicaciones. Encuentra la línea que comienza con '/LTX' y verás que ya existe una descripción para la ubicación 0. La sección conexiones, unas pocas líneas más abajo, también tiene una entrada para la ubicación 0 pero es una entrada nula.

Cambia las secciones ubicación y conexiones para que digan lo siguiente:

```
/LTX   ;Texto de Ubicaciones
/0     :Intro
El Ticket

Mientras espero el autobús en la parada, mi ticket se ha volado. ¿Puedes ayudarme a
encontrarlo?


/1
¡Estoy dentro de la bolsa!
/2
Estoy en una parada de autobús, en una carretera que va de Norte a Sur. Al Oeste, la puerta
de un parque en una verja de hierro está abierta.
/3
La hierba que piso está recién cortada. Al Norte hay un camino y un banco mientras que al
Oeste hay un estanque ornamental.
/4
Estoy en un camino de grava que va de Este a Oeste, junto a un banco del parque, al Sur hay
una zona de césped mientras que al Norte puedo ver un Quiosco de Música.
/5
Estoy en el Quiosco de Música que parece estar hecho de adornado hierro forjado pintado de
blanco. Al Sur hay un camino.
/6
El sol se refleja en la superficie del estanque ornamental, cuyas aguas se agitan con la leve
brisa. Un camino va al Norte hacia un gran árbol, mientras que al Este hay una zona de
césped.
```

```
/7
El camino lleva hacia el Sur y el Este aquí junto un gran árbol.
/8
Estoy sentado en una rama en un árbol de amplia copa, el parque se extiende debajo de mí,
al Este puedo ver la parada de autobús a través de la puerta en la verja.
;       -       -       -       -       -       -       -       -       -       -       -       -
/CON    ;Conexiones
/0                              ;Inicio del juego
        N       2
/1                              ;En bolsa
/2                              ;Parada del Autobús
        W       4
/3                              ;Junto al Banco
        N       4
        W       6
        NW      7
/4                              ;Camino
        N       5
        E       2
        S       3
        SW      6
        W       7
/5                              ;Quiosco de Música
        S       4
        SW      7
/6                              ;Estanque
        N       7
        NE      4
        E       3
/7                              ;Bajo el árbol
        U       8
        NE      5
        E       4
        SE      3
        S       6
/8                              ;Arriba en el árbol
        D       7
;       -       -       -       -       -       -       -       -       -       -       -       -
```

Puntos a tener en cuenta (montones de ellos):

a) El texto será formateado por el Intérprete cuando el juego se ejecute para asegurarse que el texto cabe en el ancho de pantalla correctamente sin partir palabras entre dos lineas.
b) No te preocupes por la ubicación 1 de momento.
c) Los textos no deben incluir ningún código de control. Cualquier carácter con un valor menor de 32 decimal se considera un código de control, incluidas tabulaciones.

d) Asegúrate de que no hay espacios al final de las líneas. La descripción de la ubicación 0 incluye 4 líneas de texto nulo. Estas líneas no deben contener ningún espacio.
e) El Compilador une las lineas de texto no nulas consecutivas con un espacio.
f) El Compilador convierte las lineas de texto nulas en retornos de carro.
g) En una pantalla de 40 columnas la pantalla de introducción debería verse así:

El Ticket

Mientras espero el autobús en la parada,
mi ticket se ha volado. ¿Puedes ayudarme
a encontrarlo?

h) Las conexiones de entrada para la ubicación 5 (por ejemplo) indican que si el jugador está en la ubicación 5 él puede teclear 'S' (o Sur) para ir a la ubicación 4 o 'SO' para ir a la ubicación 7.
i) Las palabras tras los puntos y coma son comentarios.
j) Cada dirección en la sección de conexiones, por ejemplo 'SO', debe ser definida como una palabra de movimiento en el vocabulario.
k) Para cada texto de ubicación en la sección de ubicaciones debe existir su correspondiente entrada en la sección de conexiones.
l) Hemos incluido una vía para llegar de la ubicación 0 (la introducción) a la ubicación 2. Hay una manera mejor, pero por ahora así será suficiente.

Comprueba estas entradas a conciencia con las figuras 1 y 2 hasta que estés seguro de que son correctas, graba entonces TUTORIAL.SCE y sal a CP/M para que podamos probar el Compilador. Si usas EDIT, nuestro sencillo editor de texto, con CTRL+K grabas el archivo y sales a CP/M.

Ahora vamos a probar el Compilador. Si asumimos que PAWCOMP.COM y TUTORIAL.SCE están en la unidad por defecto, debes escribir:

PAWCOMP TUTORIAL

para compilar la aventura como está en este momento. No hemos incluido ningún error deliberadamente así que deberías obtener un mensaje diciendo “Compilation ends OK” y un archivo llamado TUTORIAL.PDB. Si obtienes algún error o aviso, tendrás que corregirlo y recompilar el archivo Fuente. Nota que puedes redireccionar la salida de la compilación a un archivo de disco, si lo prefieres, indicando PAWCOMP TUTORIAL P, que creará TUTORIAL.PRN en la unidad por defecto o PAWCOMP TUTORIAL Pd que generará d:TUTORIAL.PRN. Si te encuentras realmente atascado con los errores en el archivo Fuente, el archivo TICKET.SCE puede ayudarte, pero fíjate que éste archivo contiene la aventura tutorial completa.

Nota. Un error extremadamente común, que puede originar problemas aparentemente insondables, es insertar una linea en blanco.. Particularmente una linea en blanco entre el final de una sección del archivo Fuente y la cabecera de la siguiente, por ejemplo sobre las lineas /VOC, /PRO, etc. O una línea en blanco en cualquier parte de la sección /CTL, lo que pone a PAW realmente molesto.

Jugando el Juego

Ahora podemos hacer uso por primera vez del Intérprete (y ver si lo has instalado correctamente). Asumiendo que PAWINT.COM y TUTORIAL.PDB están en la unidad por defecto, debes escribir:

```
PAWINT TUTORIAL
```

La pantalla debe limpiarse y en la parte superior debería poner algo como:

```
El Ticket

Mientras espero el autobús en la parada,
mi ticket se ha volado. ¿Puedes ayudarme
a encontrarlo?

Ahora qué?
>
```

Si no es así, debes reinstalar el Intérprete y probar de nuevo. Si tu Intérprete está instalado correctamente podemos continuar.

La línea de entrada se usa para introducir órdenes que PAW interprete como cosas para hacer – de acuerdo con la información que introdujiste al escribir el juego. De momento sólo hemos indicado dónde llevarnos al introducir ciertas direcciones. Así que prueba a empezar el juego escribiendo NORTE o N.

La pantalla se limpiará y la descripción de la ubicación 2 aparecerá – si no lo hace probablemente tienes la entrada en la sección de conexiones equivocada. No te preocupes, puede salir del Intérprete tecleando QUIT (que es un comando que PAW conoce para empezar) y respondiendo Y; quieres salir y N; no quieres, para probar de nuevo; puedes entonces corregir la entrada, recompilar y probar otra vez.

Necesitamos mencionar ahora los diagnósticos. Si pulsas RETURN/ENTER (lo que tengas en tu teclado) sin escribir nada antes, el Intérprete cambiará a modo diagnóstico y verás una línea como ésta:

Flag 38=2 ?

que muestra el valor del flag 38. Hay dos cosas que aprender aquí. Primera, pulsando RETURN/ENTER de nuevo (sin escribir nada delante) hará que el Intérprete salga del modo diagnóstico. Segunda, que el flag 38 siempre contiene el valor de la ubicación actual, en el ejemplo de arriba, 2.

Ahora puedes intentar moverte entre ubicaciones, probando los posibles movimientos (toma nota de los que estén incorrectos y poder corregirlos).

Jugando el Juego.

Podrías querer probar también algunos de los otros comando que PAW conoce, por ejemplo R o REDESCRIBE mostrará la descripción de la ubicación de nuevo – lo cual es muy útil si se ha mostrado un montón de texto y se ha perdido la descripción. I o INVENTORY listará los 'objetos' que lleves, llevarás un objeto al principio, pero no serás capaz de hacer nada con él.

Probablemente te mueras por ver al Parser en acción (por qué no?) así que si vuelves a la parada del autobús e introduces la siguiente línea, harás una visita por todo el juego.

> GO WEST THEN NORTH THEN SW THEN UP AND DOWN THEN SOUTH AND EAST THEN NORTH THEN EAST. INVENTORY

Por cierto, W.N.SW.U.D.E.N.E.I tiene el mismo efecto, pero no es tan impresionante...

Volviendo a la parte aburrida, QUIT desde el juego como vimos anteriormente y así podremos seguir con el siguiente capítulo de esta parte.

Objetos

Un objeto es todo aquello que el jugador puede manipular dentro del juego, por ejemplo, una manzana que se podría comer, una llave con la que podría abrir una puerta, o una mochila para llevar la llave y la manzana.

En nuestro juego sencillo tenemos los siguientes objetos (de los cuales no todos tienen una función en el juego final).

Objeto 0	Una linterna encendida.
Objeto 1	Una bolsa.
Objeto 2	Un sandwich.
Objeto 3	Una manzana.
Objeto 4	Un ticket.
Objeto 5	Un correa.
Objeto 6	Un abrigo.
Objeto 7	Una linterna apagada.

Fíjate que la linterna es de hecho dos objetos diferentes que podemos intercambiar si el jugador la enciende o apaga.

Comienza por editar TUTORIAL.SCE otra vez y busca la línea que comienza con /OTX. Verás que ya hay una descripción par el objeto 0. Edita la sección texto de objetos para que quede así:

```
/OTX            ;Texto de Objetos
/0
Una linterna apagada.
/1
Una bolsa.
/2
Un sandwich.
/3
Una manzana.
/4
Un ticket.
/5
Una correa.
/6
Un abrigo.
/7
Una linterna apagada.
;       -       -       -       -       -       -       -       -       -       -       -       -
```

Puntos a tener en cuenta:

a) Asegúrate de que no hay espacios al final de ninguna de las líneas.

Objetos

Ahora busca la línea que comienza con /OBJ, ésto es, la sección de definición de Objetos, y edítala para que quede como sigue:

```
/OBJ    ;Definiciones de Objetos
;obj    comienza    peso    conte-  poner/  nombre      adjetivo
;núm    en                  nedor   quitar
/0      _           1       _       _       LINTERN     ENCENDI
/1      2           3       Y       _       BOLSA       _
/2      CARRIED     1       _       _       SANDWIC     _
/3      CARRIED     1       _       _       MANZANA     _
/4      8           1       _       _       TICKET      _
/5      3           1       _       _       CORREA      _
/6      WORN        3       _       Y       ABRIGO      _
/7      CARRIED     1       _       _       LINTERN     APAGADA
; -     -       -   -       -       -       -       -   -       -       -       -       -
```

Puntos a tener en cuenta:

a) Todos los _ son el carácter de subrayado.
b) Puedes utilizar TABs para separar las columnas en la sección de definición de objetos (si tu editor de texto te lo permite).
c) Debe haber el mismo número de entradas en la sección de definición de objetos que en la sección de texto de objetos.

Qué diablos significa todo eso, te oímos preguntar. Bueno, la primera columna contiene el número del objeto (y deben ser definidos en el orden correcto). La segunda columna establece dónde estará el objeto al principio del juego; al principio del juego el objeto 2 lo lleva el jugador, el objeto 6 lo lleva puesto, el objeto 5 está en la ubicación 3 y el objeto 0 no está en ningún sitio, esto es, no existe aún en el juego. La tercera columna define el peso del objeto; los objetos 1 y 6 son tres veces más pesados que los otros objetos. La siguiente columna especifica qué objetos son contenedores, es decir que pueden contener otros objetos. Hemos decidido que sólo la bolsa será un contenedor, tienes que imaginar que un abrigo no tiene bolsillos (bueno, el nuestro no tiene en cualquier caso). La quinta columna determina si el objeto puede ponerse o quitarse – en nuestro juego solo el abrigo puede ponerse y quitarse. Fíjate que cualquier objeto que comience el juego puesto debe también marcarse con Y en la quinta columna. La siguiente columna (6ª) se utiliza para especificar el nombre asociado a cada objeto. Nota que sólo los cinco primeros caracteres de cada palabra son significativos, por tanto SANDW, SANDWI, SANDWIC y SANDWICH significan la misma cosa. La última columna se utiliza para indicar el adjetivo asociado con un objeto si el nombre solo es insuficiente para distinguir un objeto individual.

Es un excelente momento para mostrarte algunos errores del Compilador, así que graba TUTORIAL.SCE e intenta compilarlo otra vez. (PAWCOMP TUTORIAL si lo has olvidado).

Cuando el Compilador encuentra un error, el número de línea del archivo Fuente y el contenido de la línea se muestran (cuando corresponda) y el número de error y el motivo del error, por ejemplo

```
344    /1      2       3       Y       _       BOLSA       _
***    ERROR 5 -  BOLSA  is not in Vocabulary
```

Esperamos que el Compilador encuentre 8 errores así que la compilación debería acabar con la línea:

*** Compilation ends with 8 ERRORS

Los 8 errores se han encontrado porque hemos usado palabras como por ejemplo BOLSA que no han sido definidas en la sección Vocabulario. Adivina que sección vamos a ver ahora.

Vocabulario

El Vocabulario es una lista de todas las palabras que PAW es capaz de reconocer en cualquier entrada que el jugador haga durante el juego. Por ello, cualquier palabra que no esté en esta sección no tendrá efecto ninguno. START.SCE contiene unas 70 palabras inglesas comunes que se necesitan para la mayoría de las aventuras.

Cada entrada para una palabra consta de hasta cinco letras que serán o bien una palabra completa, como NORTH, o las cinco primeras letras de una palabra más larga, como ASCEN(DER), un valor de palabra y un tipo de palabra (esto es, nombre, verbo, etc)

El uso de sólo cinco letras para almacenar una palabra reduce la cantidad de memoria necesaria para almacenar el Vocabulario completo, la cantidad de tecleo que el jugador debe hacer y hace PAW más rápido al buscar las palabras cuando las necesite. Cinco letras es también lo más adecuado para diferenciar la mayoría de las palabras importantes en el lenguaje Inglés unas de otras.

Empieza por editar TUTORIAL.SCE otra vez y encuentra la sección de Vocabulario - /VOC. Verás que cada entrada consta de una palabra seguida de un valor de palabra y un tipo de palabra, así

```
                N               2               noun
```

```
PAW entiende 7 tipos de palabras, que son   noun
                                            verb
                                            adjective
                                            adverb
                                            preposition
                                            pronoun
                                      and   conjugation
```

Si echas un vistazo a la primera parte de la sección de Vocabulario encontrarás todas las direcciones que usamos en la sección de conexiones (por ejemplo, N y SW). Observa que cualquier palabra con el mismo valor de palabra y el mismo tipo de palabra se tratarán como sinónimos, esto es, que significan la misma cosa, por ejemplo N y NORTH son sinónimos. El orden en que se definan las palabras en la sección de Vocabulario no afecta. Hemos optado por agrupar todos los adverbios juntos, todos los adjetivos juntos, etc...

Objetos

Necesitamos incrementar el número de Nombres introduciendo una palabra por cada uno de nuestros objetos. El primer número libre parece ser el 15, pero valores de Nombre menores de 50 tienen un significado especial ;

Los Nombres menores de 50 son nombres propios, por ejemplo nombres de gente o lugares. Más específicamente, para PAW son nombres no son afectados por los efectos de 'it'. En la frase:

GET THE SWORD AND CLEAN IT

IT (una palabra conocida como pronombre) se refiere al la SWORD, obviamente. PAW lo sabrá (mientras que SWORD sea un Nombre en el Vocabulario con un valor mayor de 49), y asumirá que lo que intentas e limpiar la espada, permitiéndote por tanto hacerlo. Pero coge la siguiente frase:

GET THE SWORD AND KILL THE ORC WITH IT THEN DROP IT.

Normalmente PAW asocia 'it' a el último Nombre usado pero mientras ORC sea un nombre en el Vocabulario con un valor de palabra menor de 50, PAW recordará 'it' como la SWORD, y ejecutará la acción correctamente. Esta característica está indicada en los comentarios junto con la mención de valores de palabra menores de 20. Éstos son Nombres que, si PAW no encuentra un verbo en la 'frase', convertirá temporalmente (o sea, que no cambia el Vocabulario) en un Verbo. El principal uso de esto es para cosas como NORTH que puede ser escrito sólo para decir GO NORTH, que en el Inglés habitual no es válido pero es común cuando juegas aventuras.

Finalmente verbos y nombres menores de 14 se asumen como palabras de movimiento – cualquier palabra que sea una dirección. Esto seguramente determine el mensaje que se mostrará si PAW no puede hacer nada con la frase que ha encontrado (esto es, determina si se muestra 'No puedo' o 'No puedo ir en esa dirección'). Nota que esta etiqueta de 'menor de 14 es un movimiento' se aplica tanto a Verbos como a conversión de Nombres.

Dado que todos nuestros objetos son 'its' les daremos valores mayores de 49 del siguiente modo:

LINTERNA	50	noun
BOLSA	51	noun
SANDW(ICH)	52	noun
MANZA(NA)	53	noun
AUTOB(US)	54	noun
TICKE(T)	54	noun
CORREA	55	noun
ABRIG(O)	56	noun

Fíjate que hay dos palabras con el valor 54. Esto hace AUTOBUS y TICKET sinónimos, así si el jugador escribe GET BUS TICKET o GET TICKET, PAW sabrá que significan lo mismo.

También necesitamos algunas palabras para describir la diferencia entre nuestras dos linternas a PAW. Las palabras que describen un Nombre se llaman Adjetivos. Necesitamos los Adjetivos extra ENCENDIDA y APAGADA del siguiente modo:

```
ENCENDIDA       10      adjective
APAGADA         11      adjective
```

Todos los valores de palabra de 1 a 254 están disponibles para cada tipo de palabra y no hay limitación en el número de palabras con el mismo valor de palabra (sinónimos), así que el Vocabulario puede llegar a ser tan grande como quieras.

Cambia la sección de Vocabulario para incluir nuestros 8 Nombres y 2 Adjetivos extra, graba TUTORIAL.SCE y compílalo otra vez. Corrige cualquier error y recompila hasta que no haya errores o avisos.

Otra partida...?

Muy bien, vamos a probar el Intérprete otra vez (PAWINT TUTORIAL) y escribe NORTH para ir a la Parada del Autobús. Primero, un poco más acerca de los diagnósticos (puede entrar y salir del modo diagnóstico haciendo una entrada nula, esto es, sólo pulsando RETURN). Hazlo ahora... (por favor).

PAW contiene 256 de eso que se conoce como 'flags'. Cada flag puede contener un número de o 0 255 y se usan para indicar (en inglés flag) el estado de alguna parte del juego. Por ejemplo, podrías decidir que cuando el flag 11 esté puesto a 1 signifique que la puerta del parque está cerrada., y cuando esté puesto a 0 esté abierta. Veremos ejemplos de como establecer flags en la siguiente sección.

PAW tiene reservados varios flags para indicar cosas específicas (los flags 0 a 10 y 29 a 59 exactamente). El valor mostrado es el contenido del flag 38 que PAW conoce como ubicación actual:

```
Flag    38=     2 ?
```

Puedes ver los valores de otros flags escribiendo su número antes de pulsar RETURN otra vez. Prueba con 100 para ver el valor del flag 100, lo que mostrará:

```
Flag    100=    0 ?
```

Una característica muy potente te permite establecer el valor de un flag poniendo = delante del número. Prueba =10 RETURN y la línea se mostrará así:

```
Flag    100=    10 ?
```

Objetos

El flag 100 no hace nada en nuestro juego y su valor no tiene importancia, pero si decides practicar tu mismo NO cambies los valores de otros flags por el momento o puedes obtener algunos graciosos efectos si lo haces sobre un flag que sí sea importante. Vuelve a la línea de entrada cuando hayas terminado (pulsa RETURN) y así veremos más puede hacer PAW.

Ahora deberíamos ser capaces de manipular los objetos del juego. De momento, la bolsa estará en la Parada del Autobús con nosotros. Llevaremos el sandwich, la manzana,la linterna apagada y el abrigo puesto.

Utiliza los diagnósticos para ver el valor del flag 1. Éste tiene el valor 3, que es el número de objetos llevados, pero no puestos. Vuelve a la línea de entrada y escribe GET BOLSA. PAW mostrará el mensaje “I now have the bolsa.”, que se conoce como auto reporte (PAW automáticamente reporta cualquier acción que lleve a cabo). Este comando ha provocado que la posición actual de la bolsa haya cambiado de ubicación 2 (la parada del autobús) a 'ubicación' 254 (llevado). Fíjate que no se ha cambiado nada en la sección de definición de objetos de la Base de Datos, sino en una copia de ella que se hizo al comenzar el juego. Si miras el valor del flag 1 otra vez (fíjate como el flag que miraste la última vez se muestra cuando reseleccionas diagnósticos) verás que se ha incrementado a 4.

Ahora prueba REMOVE ABRIGO y el reporte “I can't remove the abrigo, my hands are full” se mostrará. Esto es porque PAW inicialmente (por defecto, en otras palabras) permite al jugador llevar sólo cuatro objetos al mismo tiempo. Esta situación debe prevenir al jugador de quitarse ropa etc (realmente remove es cambiar objetos de posición de la ubicación 253 a la ubicación 254). Prueba DROP BOLSA y después REMOVE ABRIGO otra vez – esta vez deberías poder hacerlo. Mira de nuevo el flag 1 y descubrirás que aún es cuatro – esto es porque quitarse el abrigo ha incrementado el número de cosas que tienes en tus 'manos'.

Intenta lo siguiente y mira si puedes adivinar que pasará:

GET BOLSA

REMOVE ABRIGO

WEAR ABRIGO

GET MANZANA

GET TICKET

Nota que todos, excepto el último reporte, mencionan los objetos por su nombre. Esto es porque estaban a la vista y el jugador debería saber que existen. Pero para un jugador que aún no conociera el juego, el ticket aún no había sido encontrado y mencionarlo podría implicar que existe o que hay sólo uno en todo el juego, dándonos por tanto una pista.

Si intentas poner cualquier cosa en la bolsa descubrirás que PAW en realidad deja ese objeto. Esto es porque aún no le hemos dicho a PAW qué puede ser puesto en la bolsa que es un contenedor. El próximo capítulo tratará este asunto.

Finalmente encontraremos un error en nuestro juego: escribe GET PUERTA que mostrará “There isn't one of those here.” No debería decir eso porque la descripción dice que hay una puerta aquí.

El problema surge porque aunque le dijimos a PAW sobre la manzana, el sandwich, la linterna y todo eso, no le dijimos nada de la puerta. Si usas GET (o DROP, WEAR y REMOVE) con cualquier palabra que no esté en el Vocabulario PAW asume que es un objeto que 'no está aquí'. Por supuesto, una vez que la palabra está en Vocabulario, PAW sabrá si no es un objeto (si no hay entrada para la palabra en la sección de definición de objetos) y reportará “I can't do that”, que es lo correcto.

Por tanto edita TUTORIAL.SCE e inserta las siguientes entradas de Vocabulario:

PUERTA	57	noun
BARRO(TES)	58	noun
HIERB(A)	59	noun
CAMIN(O)	60	noun
BANCO	61	noun
ESTAN(QUE)	62	noun
QUIOS(CO)	63	noun
MUSIC(A)	63	noun
ARBOL	64	noun
RAMA	64	noun
COPA	64	noun

Fíjate en como todas las formas con las que el jugador pueda referirse al árbol están atendidas. No tenemos intención de permitir la manipulación de las hojas o la rama, pero si quisieras podrías darles valores de palabra distintos – esto es una importante consideración de diseño.

Podrías usar de nuevo el Intérprete para asegurarte de que GET PUERTA ahora ya produce la respuesta correcta -no olvides recompilar TUTORIAL.SCE si lo haces, o el error permanecerá allí. Ésto sucede porque el archivo de Base de Datos refleja el estado de TUTORIAL.SCE la última vez que se compiló.

Hasta ahora hemos visto: creación de ubicaciones y conexión entre ellas, creación y descripción de objetos, asignarles una palabra en el Vocabulario, un punto de partida en el juego, peso relativo, flag de si se puede poner (y quitar) y si es un contenedor. El siguiente capítulo trata de la creación de problemas y caracteres para hacer el mundo del juego un sitio más interesante permitiendo al jugador hacer cosas.

Procesos y Respuestas

Vamos con la sección de PAW que permite la creación de problemas y personajes.

La Tabla de Respuestas

La Tabla de Respuestas es una forma especial en que PAW se refiere a una tabla de procesos. De hecho la Tabla de Respuestas es la Tabla de Procesos 0. Cuando decimos Tabla de Respuestas queremos decir Tabla de Procesos 0. Una Tabla de Procesos puede imaginarse como un sencillo lenguaje de programación secuencial (una instrucción por vez), los comandos que pueden llevarse a cabo se llama 'CondActs' porque pueden dividirse principalmente en dos grupos: Condiciones y Acciones.

Anteriormente mencionamos que el Parser de PAW divide las sentencias en frases, que son organizadas en lo que se conoce como LS (Sentencia Lógica). En el caso de direcciones como NORTH (que son LS por si mismas) utiliza la sección de conexiones para descubrir a dónde (si es posible) debe moverse el jugador. Antes de que lo haga sin embargo lleva a cabo una comprobación con la Tabla de Respuestas para ver si la tabla contiene una entrada para manejar esa LS, es decir, dar una respuesta a parte o toda la orden que escribió originalmente el jugador.

Cada frase posible que el jugador puede escribir y por tanto cada LS que tu juego responderá, tendrá una entrada correspondiente en la Tabla de Respuestas, excepto para aquellos movimientos que estableciste en la sección de Conexiones.

La parte más importante de una LS es el Verbo, éste muestra el propósito de la LS, el siguiente más importante es el primer Nombre que muestra el sujeto de la LS, por ejemplo en GET MANZANA, GET es el propósito y MANZANA el sujeto.

Mira en TUTORIAL.SCE y busca el comienzo de la Tabla de Respuestas. Comienza con la línea:

```
/PRO   0...
```

De momento ignora las otras entradas y considera sólo la primera entrada:

```
I        _        INVEN
```

las dos palabras 'I' e '_' indican el Verbo y Nombre respectivamente de la LS con la que esta entrada puede tratar. Ahora I es un nombre convertido (como vimos en la sección de Vocabulario) que significa que si es la única palabra que el jugador escribe en una frase, se convierte en el Verbo de esa LS. El subrayado (_) indica que el Nombre no es importante en esta entrada. Lo que ésto quiere decir es que si el jugador escribe I solamente, PAW lo confrontará con la primera entrada de la Tabla de Respuestas y tratará esa entrada como se describe ahora. Para llevar a cabo la entrada, PAW ejecuta cada uno de los CondActs (órdenes) de la lista que sigue. Ahora, la primera entrada contiene sólo un condact;

INVEN es una acción (la parte act de la palabra condact). Es una acción porque lleva a cabo el acto de listar los objetos que el jugador transporta y lleva puestos en la pantalla, no necesitas preocuparte de cómo INVEN lo hace, simplemente lo hace.

Cuando escribiste I (o INVENTARIO de la que es sinónimo, ¿recuerdas?) durante las pruebas del juego, fue esta entrada la que provocó que algo ocurriera porque una sentencia lógica de “I _” fue creada por el Parser, la cual PAW encontró coincidente con la primera entrada de la Tabla de Respuestas.

INVEN una vez que ha listado los objetos que llevas, indica a PAW que 'ha hecho' algo, cuando PAW recibe esto le pide al Parser otra LS, que el Parser le proporciona decodificando la siguiente frase de la entrada del jugador. PAW toma esta LS y la confronta con las entradas de respuesta y así. Este 'bucle' se muestra en la figura 3 en la forma de un diagrama de flujo que puedes seguir desde la caja llamada 'start'. El bucle es algo más complejo de lo que el diagrama te puede dar a entender, y tienes una versión completa en la Guía Técnica, aunque la figura 3 es suficiente por ahora.

Te sugerimos que releas los párrafos anteriores y estudies el diagrama hasta que comprendas la manera en que PAW actúa con las LS antes de seguir.

Vamos a considerar la segunda entrada en Respuestas:

```
GET     I       INVEN
```

Como habrás imaginado esta entrada es compatible con la frase TAKE INVENTORY (GET es un sinónimo de TAKE, I es un sinónimo de INVENTORY) que es otra manera en que el jugador podría solicitar la lista de los objetos que lleva consigo.

Saltemos unas cuantas entradas hasta:

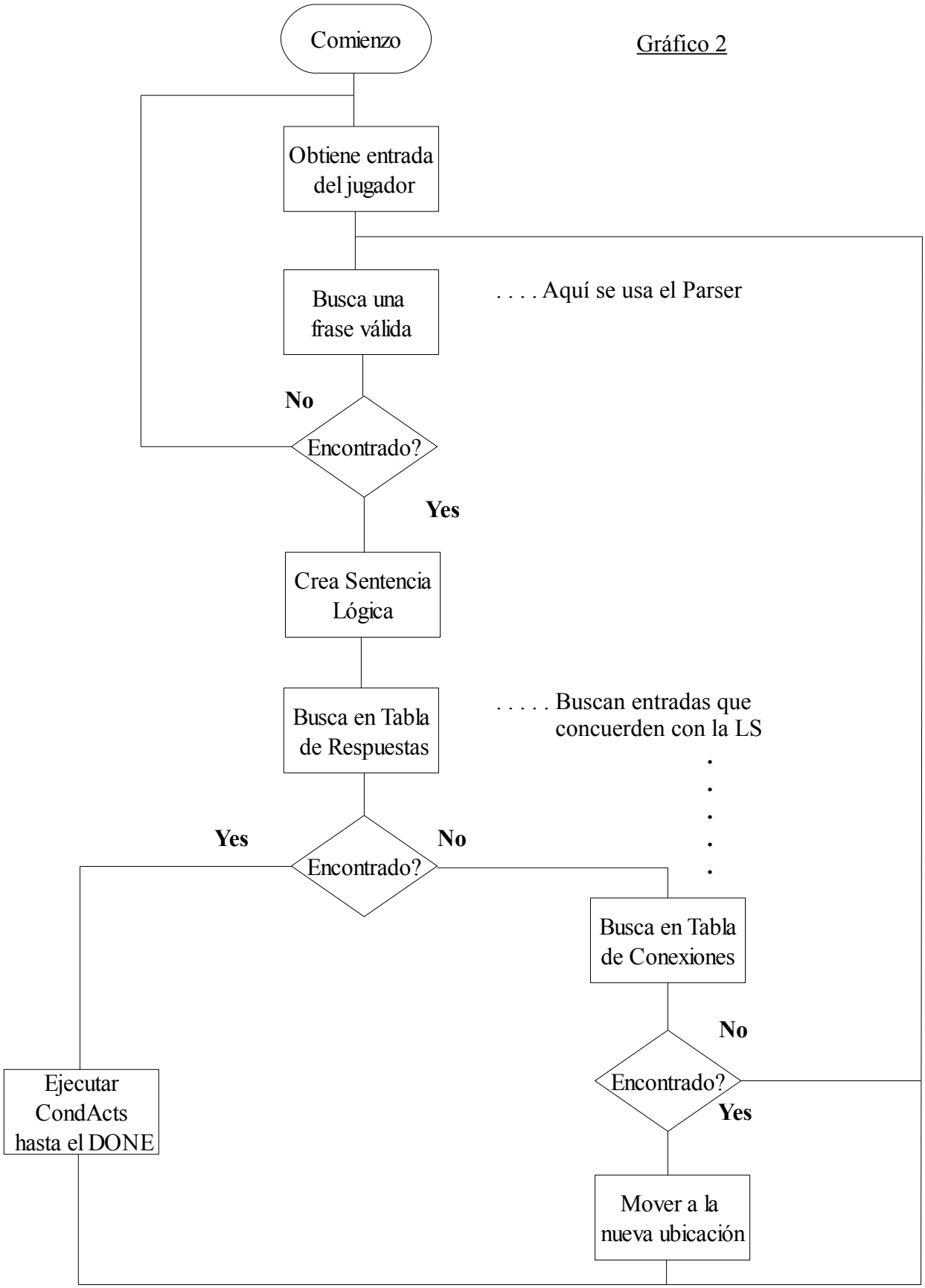
```
QUIT    _       QUIT
                TURNS
                END
```

En este caso, QUIT es un Verbo del Vocabulario, y ya que el mínimo necesario en una frase es un Verbo, si el jugador escribe sólo QUIT el Parser generará una LS “QUIT _”, y buscando en la Tabla de Respuestas PAW encontrará la entrada de arriba y llevará a cabo las condacts siguientes:

QUIT es una condición, (la parte cond de la palabra condact), no confundas el Verbo QUIT del Vocabulario con la condición QUIT, si tu quisieras hacer STOP un sinónimo de QUIT y borrar entonces la palabra QUIT del Vocabulario el jugador tendría que escribir STOP para salir del juego, pero la condición QUIT aún podría llevarse a cabo, en este caso, la entrada sería:

```
STOP    _       QUIT
                TURNS
                END
```

Procesos y Respuestas



Una condición simplemente decide si PAW debe llevar a cabo la siguiente condact de la lista. QUIT determina si la siguiente condact debe ser llevada a cabo preguntando al jugador “Are you sure?”, si él responde “NO” entonces QUIT dice a PAW que 'ha hecho' algo haciendo que PAW vaya a por otra LS (esto es, para el QUIT) esto es ligeramente diferente a la forma en que una condición funciona normalmente, como veremos más adelante. Si el jugador escribe “YES” entonces QUIT no hace nada y permite a PAW buscar la siguiente condact de la secuencia, que es TURNS.

TURNS es una acción que muestra “You have taken x turn(s).” en la pantalla donde x es el número de frases que PAW ha llevado a cabo desde que el jugador comenzó el juego. A pesar de que haya hecho algo, no le dice a PAW que deje de mirar condacts que vayan a mirar el siguiente condact END.

END es un acción especial que muestra “Would you like another go?” en la pantalla. Si el jugador escribe “YES” entonces END provoca que el juego se reinicie con todos los objetos en sus posiciones iniciales y todo eso. Si no es así, END te devuelve a CP/M.

Fíjate en que siempre debes tener una acción END en algún punto del juego (si quitaste la entrada QUIT es el caso) o no serás capaz de volver a CP/M fácilmente.

Las otras entradas que están presentes en la Tabla de Respuestas tratan con un número de otros comandos estándar que el jugador de una aventura habitualmente necesita. Los condact usados en otras entradas son tratados a continuación. Podrías preguntarte por qué estas entradas están en la Tabla y no son parte de PAW si se necesitan en todos los juegos. Aparte del hecho de que es más fácil hacerles una entrada en la tabla, tu juego podría no necesitarlos y como son una entrada de la tabla pueden eliminarse.

DESC es una acción usada por la entrada “R _” de la Tabla que provoca que PAW deje de escanear la Tabla de Respuestas y reDESCriba la ubicación actual del jugador.

SAVE y **LOAD** son dos acciones que permite al estado actual del juego grabado y recargarse de disco, la posición de juego actual incluye toda parte de información necesaria para restaurar el juego tras un LOAD para quedar exactamente en la misma posición que estaba antes del SAVE e incluye todos los valores de flags, posiciones de objetos y otra información varia. De nuevo, no confundas los Verbos SAVE y LOAD del Vocabulario con las acciones SAVE y LOAD. Podrías igualmente utilizar STORE y RECAL(L) como Verbos de tu Vocabulario pero aún usarían las acciones SAVE y LOAD en la Tabla de Respuestas. Nota que tanto SAVE como LOAD efectivamente hacen una acción DESC cuando terminan lo que significa que cualquier condact que siga será ignorada y que provocará también que cualquier frase posterior de la entrada del jugador será ignorada.

Procesos y Respuestas

RAMSAVE y **RAMLOAD** son dos acciones similares a SAVE y LOAD, excepto en que ellas utilizan un 'buffer' (área de memoria libre) para almacenar la posición del juego. Sólo puede almacenarse una posición y como es almacenada en memoria se perderá si el ordenador se apaga, esto debe quedarle claro al jugador. El número después de RAMLOAD es un parámetro y dice al condact cuántos flags restaurar desde el último RAMSAVE, lo que permite mantener las puntuaciones, etc incluso si el jugador utiliza el 'truco' de usar RAMSAVE y RAMLOAD en una parte difícil del juego. Ambas son seguidas por una acción DESC ya que no hacen como SAVE y LOAD y sólo continúan con la siguiente condact. Nota que los diagnósticos no están disponibles tras un RAMSAVE.

Si PAW agota los condacts de un lista sin que se le diga que se HA HECHO algo, llegará al final y al detectarlo continuara buscando otra LS que concuerde. También dijimos que QUIT era un poco distinta a las condiciones normales; primero, es la única condición que pide información al jugador, y segundo, dice a PAW de que se ha hecho algo si el jugador responde “NO” (no quiere abandonar la partida) lo que provoca que PAW tome una nueva LS. Una condición normal si esto falla podría seguramente hacer que PAW continúe buscando en la Tabla de Respuestas otra entrada que concuerde con la LS.

Las otras condacts que se utilizan las veremos ahora en relación a la entrada de la que forman parte. Para simplificar nuestras explicaciones consideraremos que la posición de un objeto pueda ser uno de estos cuatro lugares:

HERE: La ubicación actual del jugador (el valor del flag 38, como recordarás).

CARRIED: 'ubicación' 254, la ubicación imaginaria en que se almacenan todos los objetos que lleva el jugador.

WORN: 'ubicación' 253, la ubicación imaginaria en que se almacenan todos los objetos que lleva puestos el jugador.

NOTHERE: En cualquier otro sitio !! Esto podría incluir también la 'ubicación' 252 que es la ubicación imaginaria en que se almacenan todos los objetos que aún no 'existen'.

Toma las siguientes dos entradas de la Tabla de Respuestas:

GET	ALL	DOALL	HERE
GET	_	AUTOG DONE	

Estas dos entradas permiten al jugador GET un objeto. GET un objeto implica cambiar su ubicación de HERE a CARRIED.

Ignorando el GET ALL por un momento, vamos a echar un ojo a la entrada GET _, que como dijimos anteriormente el carácter de subrayado significa 'cualquier palabra' por lo que no hay problema en qué Nombre escriba el jugador, en la frase que contenga el GET la entrada GET _ concordará (ésto se llama atrapar la entrada). Toma la frase GET THE APPLE; THE será ignorado porque no está en el Vocabulario de PAW, por tanto, la LS será “GET APPLE”, ésto 'disparará' la entrada GET _ provocando que PAW busque condact;

AUTOG es una acción que AUTOmáticamente coGe el objeto especificado por el Nombre. Aquí es donde la sección de definición de objetos tiene efecto, AUTOG busca en la lista de definición de objetos una entrada que concuerde con el Nombre en la LS, cuando encuentra uno (APPLE en el ejemplo) conoce entonces el número de objeto al que se refiere (la manzana es el objeto 3), entonces se asegura de que la ubicación real del objeto es HERE y si es así lo cambia a CARRIED e imprime el mensaje “I now have the _.” donde el caracter de subrayado se sustituye con la descripción del objeto actual, esto es, el que AUTOG acaba de encontrar. Si no consigue encontrar una entrada, entonces hay cinco posibilidades;

1/ El jugador ha intentado coger un objeto que ya llevaba o tenía puesto en cuyo caso se muestra “I already have the _.”

2/ El jugador ha intentado coger un objeto que es NOTHERE en cuyo caso se muestra “There isn't one of those here.”

3/ El jugador ha intentado coger algo que no es un objeto pero que tiene una palabra en el Vocabulario (por ejemplo GATE en la demo) teniendo como resultado “I can't do that.”

4/ El jugador ha usado una palabra que no está en el Vocabulario lo que provoca que el Parser genere una LS de “GET _” que dispara nuestra entrada GET _ de todos modos. AUTOG asume que es un Nombre describiendo un objeto (que puede existir o no) y muestra “There isn't one of those here.”

5/ El jugador no es capaz de llevar ningún objeto más o este objeto podría provocar que se supere el límite de peso en cuyo caso se mostrará el mensaje adecuado.

Si AUTOG lo consigue entonces PAW pasará al siguiente condact DONE;

DONE simplemente dice a PAW que esta entrada ha terminado y debe ir a por otra LS.

Procesos y Respuestas

Ahora echaremos un vistazo al la entrada GET ALL, que ya habrás adivinado qué hace (intenta GET todos los objetos en la ubicación actual), así que vamos a explicar su funcionamiento;

Cuando el jugador escriba la frase GET ALL, el parser creará una sentencia lógica de “GET ALL”, que concordará con la entrada y hará que PAW mire la acción DOALL;

DOALL es una acción seguida de un parámetro que da un número de ubicación para usar. DOALL mira cada objeto de la lista de la ubicación actual buscando entradas que tengan la misma ubicación que el parámetro, cuando encuentra una mira en la sección de definición del objeto para encontrar la palabra del Vocabulario que describe ese número de objeto, es colocado en la LS actual (reemplazando así el Nombre ALL), se activa un flag para indicar que DOALL está activo y el resto de respuesta es escaneado por PAW para una entrada que coincida con la recién modificada LS. Ésta será la entrada GET _ que vimos anteriormente, que cogerá ese objeto. Una vez haya hecho ésto, PAW descubrirá que DOALL está activo y volverá a la entrada GET ALL (realmente va directamente a la acción DOALL) y permite a DOALL buscar otro objeto que genere una nueva LS y así para todos los objetos de la ubicación especificada. Cuando DOALL agota todos los objetos resetea el flag para mostrar que no está activa y decirle a PAW que consiga otra LS.

Ésta podría parecer una manera algo indirecta de conseguir esta tarea, pero si examinas las bastante similares entradas DROP, WEAR y REMOVE verás que se utiliza el mismo mecanismo para crear los cuatro comandos. AUTOD, AUTOW y AUTOR funcionan de una manera muy parecida a AUTOG mientras DOALL seguramente usa CARRIED como el parámetro para DROP y WEAR (esto es, DOALL busca todos los objetos CARRIED cuando intentas DROP o WEAR ALL!) y WORN cuando intentas REMOVE ALL.

Si esto parece un poco duro de entender no vamos a preocuparnos de momento, ya que DOALL es una de los dos más complejos condacts en PAW y seguramente se irá esclareciendo según continuemos. En este punto podrías usar el Intérprete para probar los comandos 'all' y te hará el mecanismo más claro.

Mensajes

Antes de continuar con la tabla de respuestas introduciremos algunas entradas en la sección de mensajes que serán necesarios. Así que edita TUTORIAL.SCE otra vez y encuentra la sección de mensajes (MTX no STX).

Los mensajes deben ser un soplo de aire fresco tras la discusión de la tabla de respuestas, el propósito de los mensajes es contener todo el texto que se mostrará para describir qué le está sucediendo en el juego al jugador, excluyendo los mensajes que el mismo PAW muestra (como “I can't do that.”, etc).

Estaremos de acuerdo en que el jugador querrá examinar cosas en el juego, por ejemplo EXAMINE APPLE. Ahora examinar un objeto seguramente requiera al escritor para proporcionar un mensaje que de más información sobre el objeto especificado, así en el caso de la manzana podríamos decir "The apple is crisp and green .".

Cambiar la sección de texto de mensajes para que quede como sigue:-

```
/MTX            ;Texto de mensajes
/0
La manzana es fresca y verde.
/1
Es un sandwich de queso en escabeche.
/2
El ticket lleva impreso "Compañía de Autobuses Urbanos".
/3
El banco está firmemente atornillado al suelo de hormigón.
;       -       -       -       -       -       -       -       -
```

Vamos a manejarnos con sólo cuatro objetos en el el juego demostración pero en un juego grande deberías normalmente proporcionar detalles sobre la mayoría de las cosas, incluso si ellas no tienen ningún propósito, para dar un toque de realismo que siempre hará al jugador sentirse involucrado.

Ahora volvamos a la tabla de respuestas. Vamos a coger la manzana primero; la frase que el jugador escribirá es EXAMINE THE MANZANA (o EXAMI MANZA si son vagos !) produciendo una LS de "EXAMI MANZA". Por tanto necesitaremos introducir una entrada con estas dos palabras.

Ahora sólo vamos a permitir al jugador examinar la manzana si realmente está HERE, CARRIED o WORN (la mayoría de la gente normal tiene una discapacidad que le impide ver a la vuelta de la esquina o a través de paredes!), esto es generalmente conocido como presente y puede ser comprobado usando la condición PRESENT seguida del número del objeto que estamos considerando (la manzana es el objeto 3). Si el objeto está de hecho presente entonces podemos mostrar nuestro mensaje (0) que describe el objeto, usando la acción MESSAGE seguida del número de mensaje que quieres mostrar. Finalmente lo remataremos con una acción DONE para decir a PAW que hemos completado la tarea.sta

Por tanto la entrada que necesitamos es:-

```
EXAMINE    MANZANA  PRESENT    3
                    MESSAGE    0
                    DONE
```

En el Vocabulario LOOK está definido como un sinónimo de EXAMINE y en la tabla de respuestas ya tenemos una entrada de:-

```
LOOK            -       DESC
```

Esto hace que no haya diferencia para el Compilador o el Intérprete tanto si nuestra entrada usa las palabras LOOK MANZANA o EXAMINE MANZANA. Sin embargo hace más sencillo leer y entender la tabla de respuestas si siempre elegimos la misma palabra. Por tanto cambiaremos la tabla de respuestas para introducir la siguiente entrada antes de la entrada LOOK _ existente:-

```
LOOK        MANZANA   PRESENT    3      ;Manzana presente?
                      MESSAGE    0      ;La describe
                      DONE
```

El fin de la tabla de procesos O (esto es, la tabla de respuestas) debería ahora verse así:-

```
RAMLOAD         _     RAMLOAD    255    ;Recarga todos los flags
                      DESC

LOOK        APPLE     PRESENT    3      ;Manzana presente?
                      MESSAGE    0      ;La describe
                      DONE

LOOK              _   DESC
;       -     -     -     -     -     -     -     -     -     -
```

Nuestra entrada LOOK MANZANA es un ejemplo clásico de una entrada de una Tabla de Respuestas porque si la condición PRESENT 3 falla entonces PAW continuará buscando una entrada que concuerde LS, en este caso buscará la siguiente entrada que es LOOK _, esta entrada la atrapará y describirá la ubicación actual (la acción DESC). Suponiendo que la MANZANA sí hubiera estado presente entonces PAW continuará con las condacts y mostrará nuestra descripción de la manzana (MESSAGE 0) la acción DONE dirá a PAW que vaya a por otra LS porque hemos hecho algo – ésto previene que lo capture la entrada LOOK _.

Así, de momento si el jugador intenta EXAMINE cualquier cosa excepto la manzana (o intenta EXAMINE MANZANA cuando no está presente) se le recompensará con una rica descripción de la ubicación actual. Esto es por lo que la entrada LOOK MANZANA DEBE ser especificada antes de la entrada LOOK _. Vamos a introducir las entradas para manejar el sandwich, el ticket y el banco – éstos son:-

```
LOOK        SANDW     PRESENT    2      ;El sandwich está aquí
                      MESSAGE    1      ;Lo describe
                      DONE

LOOK        TICKET    PRESENT    4      ;El ticket está aquí
                      MESSAGE    2
                      DONE

LOOK        BANCO     AT         4      ;El banco no es un objeto
                      MESSAGE    3      ;así que comprueba la ubicación
                      DONE
```

y deben introducirse antes de la entrada LOOK _.

AT es una condición, seguida de un número de ubicación, la cual se conseguirá (esto es, permitirá a PAW continuar con la siguiente condact) si el jugador está en la misma ubicación, esto se usa porque el banco no es un objeto, sino que como es parte de la descripción de la ubicación 4 siempre estará ahí!

Compila la aventura y pruébala para comprobar que puedes examinar estos cuatro objetos correctamente y que la ubicación se describe en las otras ocasiones.

Las Tablas de Procesos

Vamos a centrar nuestra atención sobre las otras tablas de Procesos.

Se estableció anteriormente que la Tabla de Respuestas era la tabla de procesos 0. Puede haber hasta 254 tablas de procesos como veremos.

Hay tres tablas de procesos (0, 1 y 2) en el archivo fuente con las que empezar, PAW las escanea como respuesta, pero, no como respuesta, las escanea no después de obtener una LS, sino:

Proceso 1 es escaneado inmediatamente después de que PAW ha descrito una ubicación. Esto permite que se imprima la información sólo una vez cuando el jugador llegue a una ubicación o cuando él solicite una redescripción.

Proceso 2 justo antes de solicitar una nueva LS desde el parser. Esto se usa para proporcionar turno de PAW durante el juego.

Hasta ahora, mientras jugábamos nuestro juego de demostración teníamos que escribir QUIT para acabar el juego. Ahora la linea de la historia original (si puedes recordar tanto) era ayudar al pasajero a encontrar el ticket antes de que llegara el autobús. Ahora obviamente deberíamos tener una entrada en respuesta a que el jugador diga GET TICKET (y si está presente) nos lleve al final del juego, por ejemplo

```
GET   TICKET      PRESENT    4      ;El ticket está aquí
                  TURNS
                  END               ;Eso es todo, amigos !
```

pero ¿no sería mucho mejor acabar el juego cuando el jugador volviera a la parada del autobús?

Lo haremos así, pero primero necesitamos un mensaje para describir la llegada del autobús, por tanto añade el siguiente mensaje a la sección de texto de mensajes:-

```
4/
Llega el autobús. Le doy el ticket al conductor que
sonríe y dice “Perdón por el retraso, espero que no haya
estado esperando demasiado”.
```

Nota que el mensaje 4 debe introducirse tras el mensaje 3. Las condiciones para el fin del juego son que el jugador esté en la parada del autobús (ubicación 2) y llevando el ticket (objeto 4). La primera condición por supuesto de ser AT 2, la otra puede comprobarse con CARRIED 4 (estas condiciones tienen nombres poco usuales...) por tanto la entrada final necesaria en Proceso 2 es:-

```
_       _       AT          2       ;en la parada del autobús
                CARRIED     4       ;con el ticket
                MESSAGE     4       ;el final
                TURNS
                END
```

esta entrada será ahora escaneada justo antes de PAW busque una nueva LS y en cuanto ambas condiciones se consigan el juego acabará independientemente de los comandos que el jugador utilice para llegar a la parada del autobús con el ticket!

Añade esta entrada a Procesos 2 ahora, esto es, al final de TUTORIAL.SCE.

Si miras en la Tabla de Procesos 1 verás que están presentes las siguientes entradas:-

```
_       _       NEWLINE
                ZERO        0       ;si hay luz...
                ABSENT      0       ;y la fuente de luz está ausente
                LISTOBJ             ;Lista los objetos

_       _       PRESENT     0       ;Si la fuente de luz está presente
                LISTOBJ             ;Lista los objetos
```

Cuando se describe cada ubicación la primera acción NEWLINE será siempre ejecutada;

NEWLINE imprime un retorno de carro. Su propósito principal aquí es asegurar que cualquier texto mostrado esté en una nueva linea porque PAW no empieza una tras mostrar la descripción de una ubicación, la Guía Técnica muestra cómo usar esto para mejorar el efecto al modificar la descripción de una ubicación para mostrar cambios en la ubicación.

Desde ahora en las dos entradas que deben considerarse como un par, su propósito final es mostrar los objetos de la ubicación actual, primero un poco de información de fondo.

PAW usa el flag cero para determinar si hay luz para que vea el jugador (esta característica no se usa de momento en nuestro juego de demostración), si no hay luz el flag tendrá un valor distinto de cero y PAW dirá “It's too dark to see anything.” en lugar de la descripción de la ubicación. En este caso los objetos que estén presentes no deben listarse.

El Objeto 0 se asume por PAW como un objeto que proporciona luz y por esto es que el objeto 0 de nuestra demo es una linterna encendida. Si ésta está presente mientras el juego está 'oscuro' (flag 0 no es cero) entonces se desentenderá de la oscuridad y así los objetos se mostrarán.

Estas dos entradas proporcionan un ejemplo de cómo usar PAW para crear una situación OR esto es, listar los objetos si hay luz OR si el objeto 0 está presente;

ZERO es la primera condición que nos encontramos que comprueba el estado de un flag. ZERO 0 se conseguirá si el flag cero contiene un 0, que significa que hay luz.

ABSENT se asegura de que el objeto 0 no está presente (contraria a la condición PRESENT – todas las condiciones tienen un opuesto, por ejemplo AT tiene un opuesto NOTAT y así.), la siguiente entrada _ _ lista los objetos si el objeto 0 (la fuente de luz) está presente así que nosotros no queremos que esta entrada se consiga (esto es, esto maneja la situación de haber luz y estar presente el objeto 0 que podría de otra manera mostrar los objetos dos veces !).

LISTOBJ lista cualquier objeto que esté presente en la ubicación actual del jugador, si no hay ninguno presente no hace nada – podría parecer un poco tonto decir “I can also see nothing.”!

Piensa que lo de arriba representa una útil característica de PAW que podrías necesitar adaptar para usar en tus propios juegos.

Perfecto, ahora revelaremos la mejor manera de ir desde la pantalla de introducción al inicio del juego en la parada del autobús;

```
_       _       AT          0       ;Inicio del juego
                ANYKEY
                GOTO        2
                DESC
```

Introduce ésto en Procesos 1 delante de las dos entradas existentes. Ésto utiliza dos nuevas condacts ANYKEY y GOTO las cuales son ambas acciones;

ANYKEY imprime “Press any key to continue.” y espera a que pulses una tecla, permitiendo entonces a PAW continuar con la siguiente condact.

GOTO es seguida de un número de ubicación y mueve al jugador a esta ubicación, efectivamente establece el flag 38 (ubicación actual del jugador) al valor dado, no hace nada si no es seguida por un DESC que diga a PAW que muestre la nueva descripción.

Esta entrada por tanto causa que se muestre la pantalla de título (cuando PAW muestra la descripción de la primera ubicación), espera por una tecla y entonces el juego comienza en la correcta ubicación.

Te podría gustar ir la sección de conexiones y eliminar la entrada para NORTH en la ubicación 0 ya que ahora no se necesita.

Compila y prueba la aventura para ver las dos entradas de arriba en acción. La siguiente entrada, estando en la ubicación 2 (la parada del autobús) 'resolverá' el juego en un intento:

GO WEST, WEST AND UP, GET THE TICKET, GO DOWN, EAST AND EAST.

Deberías entonces conseguir el mensaje de finalizado y una opción para jugar otra vez. Si no es así, comprueba las entradas en las tablas de Procesos 1 y 2.

Hagamos un descanso y volvamos a jugar con la capacidad de la bolsa para contener objetos. Pensaste que nos habíamos olvidado de eso, ¿eh? Bueno, casi lo hicimos. Ésto requerirá algunas entradas en la tabla de respuestas y vamos a permitir al jugador LOOK IN BOLSA, por lo que necesitaremos un nuevo mensaje “En la bolsa hay:”, añádelo (debería ser el mensaje 5) a la sección de mensajes. Vamos a dotar al jugador con la opción de decir PUT ALL IN BOLSA así como PUT objeto IN BOLSA. Podemos usar exactamente el mismo sistema de GET/DROP ALL que vimos anteriormente. PUT es un sinónimo de DROP (que controla DROP TICKET IN BOLSA y frases similares), por tanto la LS que debemos comprobar será DROP _ (esto es, el jugador intenta poner o dejar algo), ahora si el jugador incluye IN BOLSA como parte de la frase nosotros queremos que PAW ponga el objeto en la BOLSA. Esto significa que podemos esquivar la entrada DROP _ ya presente, y si las palabras extra están incluidas en la LS poner el objeto especificado en la bolsa.

Por tanto, la entrada que necesitamos es:-

```
DROP    _       PREP        IN      ;Pone algo en la bolsa
                NOUN2       BOLSA
                PRESENT     1       ;Está la bolsa presente?
                AUTOP       1
                DONE
```

así que introdúcela entre la entrada DROP ALL existente y la entrada DROP _. Esto muestra como podemos comprobar una LS extendida (esto es, asegurarse de que ciertas partes de la frase son las que necesitamos).

PREP es una condición que es seguida de una preposición del vocabulario. Las preposiciones son palabras que se usan antes de un Nombre para mostrar su relación con otra palabra de la frase, en este caso la condición se conseguirá si el jugador ha usado IN como parte de la frase.

NOUN2 es una condición que es seguida por un Nombre del Vocabulario. Esto se conseguirá si el jugador ha usado BOLSA en la frase. Combinado con la entrada previa, efectivamente detiene a

PAW en la búsqueda de condacts a menos que la LS fuera PUT _ IN BOLSA donde el subrayado es un objeto.

AUTOP va seguido de un número de ubicación. Anteriormente en el tutorial habíamos dejado apartada la ubicación 1 para un propósito especial, es éste, se usa como el interior de la bolsa! Así AUTOP igual que AUTOD busca en la sección de definición de objetos un Nombre que concuerde el primer nombre actual de la LS, cuando encuentra uno lo coloca en la ubicación dada, reportando "I have put the _ in the bolsa."!

La entrada DROP ALL que hay funcionará también con PUT ALL IN BOLSA, porque no se asegura de que IN BOLSA es parte de la LS y apuntará en ambas ocasiones, y en ambos casos 254 (o CARRIED) es la ubicación de la que vienen los objetos.

Ahora para un comando del tipo GET un objeto OUT OF BOLSA necesitaremos una entrada similar a la de arriba para evitar la entrada GET _ que ya existe e introducir la siguiente entrada entre las entradas existentes GET ALL y GET _:-

```
GET     _       PREP        OUT     ;Saca algo de la bolsa
                NOUN2       BOLSA
                PRESENT     1       ;Está la bolsa?
                AUTOT       1
                DONE
```

AUTOT va seguido de un número de ubicación que muestra de donde viene el objeto a TAKEOUT.

La implementación de una versión ALL del comando necesita una entrada propia, de momento GET ALL provoca un DOALL HERE (o 255), que es la ubicación actual del jugador, para ser llevada a cabo. Para poder sacar todo de la bolsa necesitamos generar todos los objetos que tiene dentro (ubicación 1) así, introducir la siguiente entrada delante de la existente entrada GET ALL:-

```
GET     ALL     PREP        OUT     ;Saca todo de la bolsa
                NOUN2       BOLSA
                DOALL       1
```

Antes de que pruebes la aventura otra vez introduce la siguiente entrada antes de la entrada LOOK _ existente. Esto permite al jugador LOOK IN THE BOLSA:-

```
LOOK    BOLSA   PREP        IN      ;Mira en la bolsa
                MESSAGE     5
                LISTAT      1
                DONE
```

LISTAT va seguido de un número de ubicación y lista cualquier objeto presente en esa ubicación, nota que si no hay objetos presentes imprimirá "nothing." así lo de arriba podría resultar:

```
En la bolsa hay:
nothing.
```

que es correcto, no como LISTOBJ que no imprime nada en esa situación.

Así que compila y prueba la aventura otra vez para asegurar que de hecho puedes PUT ALL IN BOLSA y GET objeto OUT OF BOLSA, etc.

El Pájaro

El juego del tutorial es un poco fácil de resolver así que vamos a añadir un poco de complejidad en la forma de puzzles creando dos caracteres que deambulen por nuestro pequeño mundo. Estos caracteres son llamados 'Pseudo-Inteligentes' (PSI para abreviar) porque obviamente no pueden pensar, pero debe de parecerle al jugador que sí lo hacen. Un PSI consiste principalmente en una colección de mensajes, flags y entradas en tabla de proceso, pero incluso unas pocas entradas simples se pueden crear un sorprendente efecto realista. Crear un PSI complejo de digamos un humano puede tomar algo más de elaboración, pero sigue los mismos principios que los que cogeremos con nuestros dos PSIs: un pájaro y un perro.

El pájaro complicará el escenario de la siguiente manera: cogerá el billete al principio del juego (normalmente asignarías una ubicación no utilizada para contener los objetos del pájaro, pero nosotros usaremos la localidad 252 – el objeto no existe en el juego – ya que solamente tenemos un PSI que pueda llevar un objeto). Esto significa que debes convencer al pájaro para que suelte el ticket, intentar GET resultará en el mensaje "I can't do that" y el pájaro se irá volando. El pájaro también volará entre el Quiosco de Música y la Rama del Árbol a intervalos regulares. La manera de conseguir que el pájaro suelte el ticket será dejar el sandwich en la misma ubicación. Así que vamos a hacer ese montón de trabajo primero.

Primero cambiaremos la definición del objeto 4 a:-

```
/4      _       1       _       _       TICKET      _
```

Ésto hace que el ticket no exista como objeto, lo hacemos para indicar que está en el pico del pájaro. Introduce los Nombres PERRO y PAJARO en el Vocabulario con valores de palabra 21 y 22 respectivamente. Después, introduce los siguientes mensajes, que serán necesarios.

```
/6
El pájaro suelta el ticket para picotear el sandwich..
/7
El pájaro secuestra el ticket.
/8
El pájaro me ignora.
/9
Aquí hay un pequeño pájaro.
/10
El pájaro lleva un ticket en su pico.
/11
Un pequeño pájaro se posa en el suelo.
/12
Un pequeño pájaro se posa en la rama.
/13
El pájaro ve al perro y se aleja aleteando rápidamente.
/14
El pájaro se va volando.
```

Introduciremos los mensajes para tratar con el perro más tarde tras verotra característica de PAW.

El Pájaro

En un juego grande que contenga varios PSIs y un montón de acciones de fondo, las Tablas de Procesos 1 y 2 pronto acabarán llenas de entradas haciéndose imposible ver cómo funcionan. Llega al rescate la fase destinada a las otras tablas de procesos, éstas pueden ser 'llamadas' desde las de Procesos 1 y 2 o Respuesta y usadas como una extensión de la tabla desde la que fueron llamadas. Llamar a un proceso hace que PAW guarde dónde está en ese momento y lleve la acción a la tabla indicada. Nota que cuando algo está DONE en el proceso llamado, PAW volverá a la tabla original, pudiendo por tanto conseguirse acciones bastante complejas mediante el uso de estos 'sub-procesos'. Los usuarios que programen en otros lenguajes reconocerán a éstas como 'subrutinas'.

Mientras PAW está en un sub-proceso es posible que sea llamado para actuar en otro sub-proceso más - ¿un sub-sub-proceso? y así se puede continuar con sub-sub-sub-procesos! Se pueden llevar a cabo hasta 10 niveles de subrutinas, esto se llama 'anidar' una llamada, intentar ir más allá de 10 provocará un error de tiempo de ejecución del Intérprete.

No vamos a usar nada como eso, solamente un sencillo sub-proceso. Éste contendrá todas las entradas para tratar con las actividades del pájaro.

Usaremos la Tabla de Proceso 3 para el pájaro.

El Flag 11 será un flag 'operativo' que contendrá un valor para usar en una comparación.

El Flag 12 contendrá el número de ubicación actual del del pájaro.

El Flag 5 es un flag especial que si tiene un valor distinto de 0, PAW disminuirá 1 cada vez que revisa la tabla de Procesos 2 – esto se llama un flag auto-decreciente. En este caso se usa para contar el número de 'porciones de tiempo' que han pasado en el juego, una porción de tiempo es toda una vuelta del bucle grande de la figura 3, y de momento, esto sucede antes de cada frase que escribe el jugador. El pájaro cambiará de ubicación cada tres frases, lo que creará la apariencia de acción en el juego independiente de la entrada del jugador.

Ahora, vamos a introducir las siguientes entradas, en orden tras el proceso 2, cada entrada está precedida por una explicación de su propósito y cualquier nuevo condact que use:

Primero se determina si el pájaro se va a ir volando esta vez a través de la tabla, ésto es indicado por el flag cinco si es cero (ya que empieza a contar desde 3), si el ticket está en la misma ubicación que el pájaro, será destruido (esto es puesto en la ubicación 252 así el pájaro lo 'tiene') y si el jugador está en la misma ubicación que el pájaro se le dirá que el pájaro ha cogido el billete. Nota que el pájaro sigue con su ciclo de movimientos incluso si el jugador no lo ve, en PAW un árbol ciertamente se cae incluso si no hay nadie para verlo!

```
/PRO  3      ;Pájaro
_     _      COPYOF    4     11    ;Copia la ubicación de objeto 4 (ticket) a flag 11
             SAME      11    12    ;ticket en la misma ubicación que el pájaro
             ZERO      5           ;Se va a ir el pájaro?
             DESTROY   4           ;El pájaro 'COGE' el ticket
             SAME      12    38    ;Pájaro en la misma ubicación que el jugador?
             MESSAGE   7           ;Se lo dice al jugador.
```

Fíjate que no hay acción DONE ya que queremos que PAW haga cada entrada por turno, la entrada de arriba muestra cómo se pueden mezclar condiciones y acciones para crear nuevas condiciones.

La linea /PRO 3 marca el inicio de la tabla de Procesos 3.

COPYOF es una acción seguida de un número de objeto y un flag, copia la ubicación actual del objeto especificado al flag especificado. La usamos en esta situación para ver si el ticket está en la misma ubicación que el pájaro.

SAME es una condición que compara el contenido de los dos flags, y que es conseguida si ambas son el mismo.

DESTROY es una acción que coloca el objeto especificado en la ubicación 252, la ubicación no-creados.

Ahora trataremos los dos posibles movimientos del pájaro. Si el pájaro está en el Quiosco de Música y el flag cinco ha alcanzado el cero entonces el pájaro se moverá, se pondrá el flag 5 a 3 otra vez y se dirá al jugador que el pájaro se ha ido, si ambos estaban en la misma ubicación. Y lo mismo si el pájaro está en la rama.

```
_     _     EQ        12    8     ;¿Está el pájaro en la rama?
            ZERO       5          ;¿Toca volar?
            LET       12    5     ;Mueve el pájaro al Quiosco
            LET        5    3     ;Tres frases hasta el movimiento
            AT         8          ;Está también el jugador aquí?
            MESSAGE   14          ;Dice que el pájaro ha volado

_     _     EQ        12    5     ;El pájaro está en el quiosco
            ZERO       5          ;¿Toca volar?
            LET       12    8     ;Lo mueve a la rama
            LET        5    3     ;Tres frases hasta el movimiento
            AT         5          ;Está también el jugador aquí?
            MESSAGE   14          ;Lo dice...
```

EQ es una condición que va seguida de un número de flag y un valor y se conseguirá si el flag contiene ese valor, en este caso está comprobando si el pájaro está en una ubicación específica.

LET es una acción que va seguida de un flag y un valor. Pone el flag con ese valor.

El Pájaro

Ya hemos tratado la salida del pájaro, ahora vamos a tratar su llegada, y si llega a una ubicación en la que esté el jugador, se lo dirá.

```
_    _    EQ        5    3     ;¿Acaba de volar el pájaro?
          SAME     12   38     ;¿Está en la ubicación del jugador?
          AT        5          ;¿En el Quiosco de Música?
          MESSAGE  11          ;Posado en tierra

_    _    EQ        5    3     ;¿Acaba de volar el pájaro?
          SAME     12   38     ;¿Está en la ubicación del jugador?
          AT        8          ;¿En la rama?
          MESSAGE  12          ;Posado en rama
```

Ahora si el pájaro tiene el ticket en su pico, debemos decírselo al jugador.

```
_    _    EQ        5    3
          SAME     12   38
          ISAT      4  252     ;¿El ticket ha sido creado?
          MESSAGE  10          ;Tiene el ticket en su pico.
```

ISAT es una condición seguida de un objeto y un número de ubicación y se consigue si el objeto se encuentra en la ubicación especificada.

Finalmente, si el sandwich está en la misma ubicación que el pájaro, el pájaro dejará el ticket para coger el sandwich. Esta entrada no tiene relación con el flag 5 así que será comprobada cada vez que PAW compruebe el Proceso 2, por tanto incluso si el jugador deja el sandwich <u>después</u> de que el pájaro haya llegado, la secuencia correcta aun se llevará a cabo.

```
_    _    COPYOF    2   11     ;Sandwich
          SAME     11   12     ;¿Está en la misma ubicación que el pájaro?
          ISAT      4  252     ;¿Ticket en el pico?
          COPYFO   12    4     ;Deja el ticket
          SAME     12   38     ;¿Está el jugador aquí también?
          MESSAGE   6          ;Lo dice...
```

COPYFO es una acción que copia el contenido del flag especificado a la ubicación actual del objeto especificado. También hay acciones COPYFF y COPYOO de las que seguramente adivines el propósito.

Ésto completa las rutinas de control del pájaro, pero necesitamos una entrada en el Proceso 2 para llamar a esta tabla en cada porción de tiempo, así que introduce la entrada siguiente al comienzo de la Tabla de Procesos 2:-

```
_    _    PROCESS   3            ;Pájaro
```

lo que hará que PAW ejecute nuestra tabla de control del pájaro cada vez que pase por ahí el bucle principal.

Tenemos que asegurarnos de que el pájaro empieza en la ubicación correcta y que el jugador sabe que el pájaro está ahí cuando se describe la ubicación (o empezará a ver mensajes acerca de un pájaro que va y viene sin que la descripción lo mencione). Así que corrijamos la entrada que hicimos en el Proceso 1 para que contenga un LET 12 8, lo que asegurará que el pájaro se encuentre en la rama al principio del juego. La entrada modificada debería quedar:

```
_       _       AT          0           ;Inicio del juego
                ANYKEY
                LET         12      8   ;El pájaro está en la rama (ubicación 8)
                GOTO        2
                DESC
```

Introduce la siguiente entrada al final de la Tabla de Procesos 1 para decir al jugador que el pájaro está presente y si tiene el ticket.

```
_       _       SAME        12      38  ;¿Está el pájaro en la misma ubicación?
                MESSAGE     9           ;Se lo dice al jugador
                ISAT        4       252 ;¿Ticket en el pico?
                MESSAGE     10          ;Se lo dice al jugador
```

Finalmente en la sección de respuestas introduce la entrada:

```
GET     TICKE   SAME        12      38  ;¿Está el pájaro en la misma ubicación?
                ISAT        4       252 ;¿con el ticket en el pico?
                CLEAR       5           ;Le fuerza a volar
                NOTDONE                 ;"I can't do that"
```

Ésto debería ir antes de las entradas GET ALL y previene que se produzca el mensaje "There isn't one of those here" si el pájaro está presente con el ticket.

CLEAR es una acción que va seguida por un número de flag y hace que ese flag tenga el valor 0. Ésto hará que el pájaro salga volando (podría haberse ido de todos modos) simulando su miedo a una gran mano que desciende sobre él para coger su preciada posesión.

NOTDONE es una acción similar a la acción DONE pero que engaña a PAW para que piense que no ha hecho nada y por tanto imprima el mensaje "I can't do that".

Ahora el momento de la verdad, tras probar el juego debes haber visto al pájaro entrando y saliendo volando del Quiosco de Música y la rama, juega un rato para ver que de hecho el pájaro continúa con su itinerante existencia. Después intenta dejar el sandwich en la misma ubicación. Nota que si no coges el ticket antes de que el pájaro se vaya el pájaro lo cogerá otra vez.

El Perro

El perro lo añadimos para complicar el juego un poco más. El perro simplemente seguirá al jugador a todas partes (siendo un perro obediente) y hará que el pájaro se asuste. Ahora un perro no debería ser capaz de trepar al árbol así que debemos prever que el jugador no pueda tentar al pájaro con el sandwich en la rama. Para hacer eso estableceremos que cualquier objeto que se deje en la rama caiga al suelo. El jugador será capaz de desembarazarse del perro poniéndole la correa y atando la correa al banco. Además el jugador será capaz de 'hablar' al perro lo que le proporcionará otra manera de quitárselo de encima diciéndole SIT o STAY.

Antes de examinar las entradas en Procesos y Respuestas, necesitamos controlar el perro introduciendo siguientes palabras en la sección de Vocabulario:-

TIE	34	verb
UNTIE	35	verb
SIT	36	verb
STAY	37	verb
COME	38	verb
HERE	37	noun

y los siguientes mensajes en la sección de mensajes:-

```
/15
The _ falls to the ground at the foot of tree.
/16
The dog's bright eyes stare at me with mindless love.
/17
A dog is here.
/18
The dog follows me wagging his tail.
/19
A lead trails behind the dog.
/20
The dog is tied to the bench by a lead
/21
Trustingly the dog lets me put the lead around his neck.
/22
I've tied the lead to the bench.
/23
Who should I say it to?
/24
The dog is sitting quietly.
/25
I've untied the dog from the bench.
```

Asegúrate de que incluyes el carácter de subrayado en el mensaje 15 ya que tiene un propósito especial que veremos más adelante.

No hay verdadera necesidad de hacer la rutina de control para el perro en una tabla de Procesos separada ya que solamente tiene una entrada, pero lo haremos por si quisieras ampliar el juego más adelante.

El flag 13 contendrá la ubicación actual del perro.

El flag 14 contendrá: 0 – el perro está suelto, 1 – el perro tiene la correa puesta, 2 – el perro está atado al banco, 255 – el perro está sentado tranquilamente.

La Tabla de Procesos necesaria para el perro es:-

```
/PRO  4       ;Perro
_     _       NOTSAME   13     38     ;¿Está el perro donde está el jugador?
              LT        14      2     ;¿Aún se puede mover?
              NOTAT      8            ;¿Está el jugador subido al árbol?
              COPYFF    38     13     ;Mueve al perro a la ubicación del jugador.
              MESSAGE   18            ;Dice que le ha seguido...
```

Deberías ser capaz de ver lo hacen NOTSAME, NOTAT y COPYFF, si no la Guía Técnica te ayudará si tienes problemas.

LT es una condición que se consigue si el flag especificado contiene un valor Menor Que el valor especificado.

Introduce al comienzo de la Tabla de Procesos 2:

```
_     _       PROCESS    4           ;Perro
```

Nota que ésta viene antes de la entrada del pájaro para asegurarse de que el perro se moverá a la nueva ubicación del jugador antes de que se compruebe el pájaro.

De manera similar a como hicimos con el pájaro, se requieren entradas en la Tabla de Procesos 1 para informar al jugador de la presencia del perro y éstas deberían ir antes de las entradas del pájaro:-

```
_     _       SAME      13     38     ;¿Perro en la misma ubicación?
              MESSAGE   17            ;Se lo dice al jugador
              EQ        14      1     ;¿Con correa?
              MESSAGE   19            ;Si sí se lo dice al jugador

_     _       SAME      13     38
              EQ        14      2     ;¿Perro atado al banco?
              MESSAGE   20

_     _       SAME      13     38
              GT        14      2     ;255 es mayor que 2 así que
              MESSAGE   24            ;le dice al jugador que el perro está sentado
```

ya que estás en el Proceso 1 modifica la primera entrada para que contenga un LET 13 2 (antes del GOTO) para hacer que el perro comience en la parada del autobús.

Ahora para que el pájaro se asuste del perro necesitamos una entrada extra en la tabla de Procesos 3. Eso sí, debe ir antes de la entrada que decide que suelte el ticket y después de la entrada que hace que el pájaro vuele. Así nos aseguramos que el pájaro vuele con el ticket si lo tiene y lo deje si no lo tiene. Por tanto necesitamos introducir lo siguiente como séptima entrada en Procesos 3:-

```
_       _       SAME 12      13      ;Pájaro y perro en la misma ubicación
                LET  12       8      ;Sólo puede ser el Quiosco de Música así que
                LET   5       3      ;lo mueve al la rama, tres frases
                AT    5              ;¿Está el jugador en el Quiosco?
                MESSAGE      13      ;le dice que el pájaro se ha ido.
```

El último cambio en la Tabla de Procesos es introducir un sub-proceso que llamaremos desde Respuestas para manejar el diálogo con el perro. El mecanismo es muy simple. Si el jugador incluye una frase entrecomillada (" ") en la sentencia INPUT, el parser guardará el lugar donde se encuentra y continuará descifrando la frase. Hay una acción llamada PARSE que instruye a PAW para que decodifique la cadena que el jugador ha tecleado, haciendo que esto se la LS. Solo tiene sentido hacer ésto en un sub-proceso ya que PAW tratará de concordar la nueva LS con el resto de la tabla. Introduce lo siguiente para hacer la Tabla de Procesos 5:-

```
/PRO  5       ;Hablar al perro
_       _     PARSE                    ;Convierte la cadena a LS
              MESSAGE    16            ;Frase no válida así que
              DONE                     ;el perro no te entiende

SIT     _     ZERO       14            ;¿Está el perro parcialmente atado?
              SET        14            ;Ahora sentado tranquilamente
              MESSAGE    24            ;Se lo dice al jugador (siempre que esté en
              DONE                     ;el mismo sitio que el perro) Listo entonces

COME_         EQ         14    255     ;El perro debe estar sentado
              CLEAR      14            ;Ahora está normal
              MESSAGE    18            ;El perro te sigue
              DONE

_       HERE  EQ         14    255     ;¿Está el perro sentado?
              CLEAR      14            ;Ahora está normal
              MESSAGE    18            ;El perro te sigue
              DONE
_       _     MESSAGE    16            ;Algo así.
```

Evitamos el vocabulario tan limitado que entiende el perro haciendo que mueva la cola para la mayoría de las cosas!

PARSE permitirá a PAW continuar buscando condacts si no se encuentra una frase válida, se cuidadoso aquí ya que la LS actual Sentencia Lógica puede ser un poco enrevesada (esto es, el

parser intentará sacar algún significado de todos modos) así que normalmente debes introducir algún mensaje como "They didn't seem to understand" o algo similar y un DONE para volver a la acción anterior. Si se forma una LS válida PAW empezará a buscar las siguientes entradas para concordar como con Respuesta. PARSE sólo debería usarse en un sub-proceso llamado desde Respuestas, no tiene ningún significado en otra tabla.

Nota cómo las entradas COME y HERE se hacen cargo de una cantidad de frases que el jugador podría probar para llamar al perro otra vez habiendo hecho que se siente.

La última entrada coge todas las LS válidas que estuvieran en la cadena y para las que el perro no tenga ninguna respuesta específica.

Ahora a la Tabla de Respuestas para permitirá introducir las entradas extra para controlar el habla y el dejar caer objetos en el árbol.

La primera de todas es la entrada que hace que los objetos que se dejen en el árbol caigan al suelo, ahora ésto debe ir entre la entrada que se encarga de poner objetos dentro de la bolsa y la entrada normal DROP _.

```
DROP _      AT          8     255    ;¿Está el jugador en la rama?
            WHATO                    ;Ya hablaremos de ello!
            LT         51     255    ;¿Es un objeto válido?
            EQ         54     254    ;¿Es un objeto que llevas?
            MESSAGE    15            ;Ahora está debajo del árbol.
            PUTO        7            ;Puesto allí
            DONE
```

Esto es un ejemplo de cómo crear una acción automática tuya propia, como AUTOG y similares.

WHATO es una acción que mira el primer Nombre de la LS actual en la sección de defnición de objetos convirtiéndola en un número de objeto. Este número se pone entonces en el flag 51. El flag 51 siempre contiene el número del último objeto al que PAW ha hecho referencia y siempre que está activo los flag asociados 54 a 57 están también activos. El flag 54 contiene la ubicación actual del objeto.

PUTO es una acción que cambia la ubicación del objeto actualmente referenciado por la que se especifique.

El Mensaje 15 contiene un carácter de subrayado. Siempre que PAW encuentra un carácter de subrayado en un texto (sea mensaje o ubicación) lo reemplaza con el objeto actual siendo el mensaje cambiado para adaptarse al objeto con el que se está tratando actualmente.

Ahora una entrada relativamente sencilla para que se encargarse de PUT LEAD ON DOG y ésta debería ir tras la entrada DROP ALL:-

```
DROP LEAD PREP        ON          ;Se asegura de que no sea un DROP LEAD
          NOUN2       DOG
          CARRIED      5          ;El jugador tiene la correa
          SAME        13   38     ;está en la misma ubicación que el perro
          LET         14    1     ;El perro ahora tiene la correa puesta
          DESTROY      5          ;Así no la tiene el jugador
          MESSAGE     21          ;Se lo dice.
          DONE
```

Las entradas que siguen introducen un nuevo concepto otra vez, la modificación de la LS actual. Queremos que el juego entienda TIE DOG TO BENCH y TIE LEAD TO BENCH como la mism cosa, ahora, LEAD y DOG tienen diferentes valores de palabra, así que la entrada TIE DOG convierte el Nombre en LEAD (55) y permite a PAW llevar a cabo la entrada TIE LEAD! Un sistema similar se utiliza para UNTIE. Introduce las siguientes entradas en la Tabla de Procesos 0:-

```
TIE    DOG  LET          34    55     ;Convierte DOG en LEAD

TIE    LEAD PREP       TO
            NOUN2      BENCH
            AT           4          ;Donde está el banco
            SAME        13   38     ;el perro está aquí
            EQ          14    1     ;con la correa puesta
            PLUS        14    1     ;ahora atada al banco
            MESSAGE     22          ;Se lo dice al jugador.
            DONE

TIE    _    NOTDONE                 ;Asegura un I can't

UNTIE DOG   LET          34    55     ;Convierte DOG en LEAD

UNTIE LEAD AT            4          ;Donde está el banco
            EQ          14    2     ;perro atado a él
            CLEAR       14          ;Ahora está suelto
            MESSAGE     25          ;Se lo dice al jugador.
            CREATE       5          ;Vuelve a crear la correa
            GET          5          ;Intenta cogerla y lo consigue.
            DONE

UNTIE _     NOTDONE                 ;Asegura un I can't
```

El NOTDONE se asegura de que PAW reporte “I can't do that”si intentas TIE o UNTIE algo que no sea la correa/perro.

PLUS es una acción seguida por un número de flag y un valor. EL flag se incrementa por el valor. Si el resultado excede 255, el flag se establece a 255.

CREATE es una acción que va seguida de un número de objeto. Hace que el objeto aparezca en la posición donde está el jugador.

GET es una acción que va seguida de un número de objeto. Intenta coger el objeto especificado.

Usamos estas acciones en vez de simplemente poner el objeto en 254 ya que cualquier problema debido al peso y/o al número de objetos que se lleven será reportado.
Finalmente las entradas para poder hablar al perro, hemos incluido también las entradas necesarias para permitirte hablar al pájaro - él te ignorará! Éstas pueden ir también al final de la Tabla de Procesos 0:-

```
SAY   DOG   SAME      13    38    ;Está aquí
            PROCESS    5          ;Alguien que haga el trabajo
            DONE

SAY   BIRD  SAME      12    38    ;Pájaro presente
            MESSAGE    8
            DONE

SAY   _     MESSAGE   23          ;Quién?
            DONE
```

Nota que nos aseguramos de que se especifique preposición TO - ésto permite al jugador acortar su orden si quiere. Como regla general no hagas una comprobación de LS extendida a menos que sea necesario para diferenciar entre dos frases similares.

Ahora compila y prueba la aventura.

Como prueba final las siguientes entradas deberían ahora funcionar en las situaciones indicadas, te mostrarán parte de la potencia que el parser puede proporcionar a tus juegos.

Cuando estés en el camino cerca del banco del parque con la correa y el perro prueba:
PUT LEAD ON DOG AND TIE IT TO THE BENCH

Luego para desatarlo;
UNTIE DOG

Si estamos arriba del árbol con la bolsa, prueba:
PUT ALL IN THE BAG AND DROP IT. GO DOWN AND LOOK IN BAG

Para hacer que el perro se siente;
SAY TO DOG “SIT”

y para hacerle venir;
ASK DOG TO “COME HERE”

Hazlo tú mismo

Aquí hay algunos puntos que podrías mejorar en el juego de demostración para que sirvan de práctica en el uso del sistema.

1/ EXAMINE debería responder a todos los objetos aunque sea con una respuesta general como "I see nothing special about the _." . Pista: para que el uso de LOOK no se pierda en sí mismo podrías usar una condición LT 34 255 antes de lanzarla (esto es, asegurarse de que se ha especificado un nombre.

2/ El pájaro realmente debería volar si intentas GET SANDWICH mientras el pájaro está presente. Esto es, se supone que esta picoteando el sandwich y un normal pájaro se iría.

3/ UNTIE _ y TIE _ deberían tener un mensaje del estilo "Tie what to what?", NOTDONE era una salida bastante fácil.

4/ ¿Cómo manejarías al jugador que escribiera PUT objeto IN BAG cuando la bolsa no está presente? De momento, en vez de eso el juego deja el objeto, ¿por qué?

5/ Aún no se ha hecho nada con la linterna, las siguientes entradas te permitirán encenderla y apagarla (también necesitarás TURN como Verbo en el Vocabulario):

```
TURN TORCH      PREP        ON
                CARRIED     7
                SWAP        7       0
                OK

TURN TORCH      PREP        OFF
                CARRIED     0
                SWAP        0       7
                OK
```

Mira los condacts extra de la Guía Técnica y lee el capítulo sobre claridad y oscuridad – quizá se podría crear un sótano bajo el Quiosco de Música. El movimiento debería ser comprobado en la Tabla de Respuestas con una entrada del tipo: (suponiendo que 9 sea la nueva ubicación).

```
DOWN        _       AT      5       ;¿Está el jugador en el Quiosco de Música?
                    SET     0       ;Flag 0=255=Oscuridad!
                    GOTO    9       ;Nueva ubicación
                    DESC
```

No olvides una entrada para UP que limpie el flag!

6/ ¿Qué ocurre si el jugador escribe CLIMB TREE o CLIMB UP TREE y cuál la mejor manera de comprobar ésto?. Pista: Solamente hay una cosa a la que se puede subir en esa ubicación.

Fin del viaje

Esperamos que el tutorial de arriba te haya proporcionado una buena percepción de algunas de las potentes características de el Professional Adventure Writer. Ahora es el momento de que aumentes tus conocimientos sobre el sistema usándolo! La Guía Técnica te proporcionará una especificación exacta de todas las cosas que contiene PAW y junto con los intentos con él en varios temas, formará una esencial – y algo pesada – lectura cuando escribas tus propios juegos.

El archivo TIMING.SCE (cuando hayas compilado) puede utilizarse para el 'ajuste fino' de tu instalación en el Intérprete. Te ayudará a establecer los valores de PAUSE y TIMEOUT que encontramos al inicio de este manual.

Finalmente encontrarás un pequeño juego en forma de fuente en el disco llamado “TEWK”. Examinarlo te proporcionará algunas ideas más para dar a tus juegos una apariencia individual.

¿Qué debería hacer ahora?
DIVERTIRTE!!
OK
Tim Gilberts – Enero 1988
PAW GRAPHICS

Apéndice A: EDIT un sencillo Editor de Texto

Antes de que puedas utilizar el Editor de Texto éste necesita ser instalado. El programa de Edit se llama EDIT.COM, el programa de instalación se llama EDITINST.COM y la información de la instalación se guarda en un archivo llamado EDIT.INS. Para instalar el Editor de Texto escribe:-

EDITINST

el archivo EDIT.INS será leído y en tu pantalla se verá algo así:-

---

```
Columns             is set to 90
Lines               is set to 31
Clear screen        is set to '1B451B48' (hex)
Print at            is set to '1B59' (hex)
Row before column   is set to Y
Col offset          is set to 32
Row offset          is set to 32

Enter   A to change Columns
        B to change Lines
        C to change Clear screen
        D to change Print at
        E to change Row before column
        F to change col offset
        G to change row offset
or      H to Exit
Choose A-H?
```

El Editor necesita saber el número de columnas en la pantalla, el número de lineas en la pantalla, cómo limpiar la pantalla y cómo posicionar el cursor en una fila y columna en concreto en la pantalla. Nota que los códigos de Clear screen y Print at tienen que introducirse en hexadecimal. Los valores necesarios para Columns, Lines y Clear screen son los mismos que se necesitan para el Intérprete así que no deberías tener problema con ellos, pero recuerda que el código de Clear screen debe limpiar la pantalla y clocar el cursor en la esquina superior izquierda de la pantalla. Las otras variables pueden necesitar algunas explicaciones más. Para situar el cursor en un sitio particular de la pantalla el controlador de pantalla necesita recibir una secuencia de códigos particular que le indique que los siguientes valores representan una fila y una columna – éste el el código 'Print at' también conocido como el código de movimiento de cursor o secuencia de introducción de posición del cursor. El controlador de pantalla puede requerir el número de fila antes del número de columna (esto es, con Row before column puesto a 'Y') o viceversa. Cuando el controlador de pantalla esté informado de la fila y columna puede necesitar un offset par añadir a cada uno. Los valores requeridos para el offset son aquellos que posicionan el cursor en la esquina superior izquierda de la pantalla (valores comunes son 0, 1 y 32)

Algunos ejemplos de valores son:

Ordenador	BBC con Z80 (2º procesador)	PCW8512 CP/M 3	CPC464 CP/M 2	CPC6128 CP/M +
Modo pantalla	MODE 3	24x80 off	MODE 2	MODE 2
Columns	80	90	80	80
Lines	25	31	25	24
Clear screen	'0C'	'1B451B48'	'0C'	'1B451B48'
Print at	'1F'	'1B59'	'1F'	'1B59'
Row before column	N	Y	N	Y
Col offset	0	32	1	32
Row offset	0	32	1	32

Cuando creas que los valores son correctos teclea 'H' para salir y se te preguntará 'Do you want update the file on disc?'. Responde Y si has hecho algún cambio.

Para probar si el Editor se ha instalado correctamente escribe:-

EDIT EDIT.KEY

que arrancará el Editor y cargará el archivo EDIT.KEY. Tu pantalla debería mostrar algo como lo siguiente con el cursor en la esquina superior izquierda de la pantalla:-

EDIT.KEY

The Inputs allowed are	ASCII		ie value	32-126
	RETURN	CTRL M	ie value	13
	Cursor up	CTRL E	ie value	5
	Cursor down	CTRL X	ie value	24
	Cursor left	CTRL S	ie value	19
	Cursor right	CTRL D	ie value	4
	Page up	CTRL R	ie value	18
	Page down	CTRL C	ie value	3
	Top of file	CTRL T	ie value	20
	End of file	CTRL B	ie value	2
	Goto line	CTRL G	ie value	7
	Delete	DEL	ie value	127
	Delete line	CTRL Y	ie value	25
	Exit	CTRL K	ie value	11

Free:nnnnn Line:1 Col:1

Si no, deberás reinstalar el editor.

Para salir del Editor teclea CTRL K, esto es, manteniendo pulsada la tecla CTRL y pulsando K. Se te preguntará 'Save file (Y/N)?' y probablemente querrás contestar N.

Si lo has instalado correctamente verás que la línea inferior de la pantalla es una línea de estatus que te dice cuánto espacio libre hay y dónde se encuentra el cursor. El archivo que estás editando te muestra la entradas que el Editor te aceptará, por ejemplo CTRL X (esto es, manteniendo pulsada la tecla CTRL y pulsando X) para mover el cursor hacia abajo. Nota que en alguna máquinas es posible programar las teclas de cursor para mover el cursor, por ejemplo bjao CP/M 3/+ el comando SETKEYS puede usarse así:

```
SETKEYS KEYS.WP
```

que configura las teclas de cursor para producir los códigos de control de Wordstar – que son las que EDIT utiliza. SETKEYS KEYS.CCP restaurará el teclado después de que salgas.

En Uso

El editor de texto es ejecuta usando el comando 'EDIT nombrearchivo' donde nombrearchivo es el nombre de un archivo que puede o no existir. Si el archivo existe, se cargará en memoria. Mientras el archivo se lee cualquier carácter TAB que se encuentre se expandirá (esto es, se convertirá en espacios), cualquier línea que sea demasiado larga para mostrarse en pantalla se truncará y si el mismo archivo es demasiado grande se truncará.

La longitud de una línea está limitada al ancho de la pantalla menos uno, esto es, 79 para una pantalla de 80 columnas. Ésto es porque cada línea contiene un carácter de fin de linea al final. Si intentas introducir una línea más larga que ésto, obtendrás un mensaje de error 'Too long'.

The Inputs allowed are	ASCII		ie value	32-126
	RETURN	CTRL M	ie value	13
	Cursor up	CTRL E	ie value	5
	Cursor down	CTRL X	ie value	24
	Cursor left	CTRL S	ie value	19
	Cursor right	CTRL D	ie value	4
	Page up	CTRL R	ie value	18
	Page down	CTRL C	ie value	3
	Top of file	CTRL T	ie value	20
	End of file	CTRL B	ie value	2
	Goto line	CTRL G	ie value	7
	Delete	DEL	ie value	127
	Delete line	CTRL Y	ie value	25
	Exit	CTRL K	ie value	11

por ejemplo para mover el cursor al final del archivo debes mantener pulsada la tecla CTRL y pulsar B. los números mostrados arriba, por ejemplo 5 para cursor arriba, son para que te sirvan de referencia por si tienes teclas de cursor en tu teclado y quieres configurarlo para producir los códigos correctos para el Editor, por ejemplo la tecla cursor arriba debería ser configurada para producir un 5.

Cuando pulsas CTRL K para salir se te preguntará si quieres grabar el archivo que has editado. Si contestas 'Y' y estabas editando un archivo llamado TICKET.SCE, ésto es lo que ocurrirá:-

1. Cualquier archivo existente llamado TICKET.$$$ se borrará.
2. El archivo editado se grabará a uno llamado TICKET.$$$.
3. Si el archivo TICKET.SCE ya existe en el disco cualquier archivo existente llamado TICKET.BAK se borrará y TICKET.SCE se renombrará a TICKET.BAK.
4. El archivo TICKET.$$$ se renombra a TICKET.SCE.

Si en algún punto el directorio del disco o el mismo disco se encuentra lleno el Editor intentará hacer algo más de sitio y lo volverá a intentar. Si aún hay insuficiente espacio te dará la opción de cambiar de disco. Si estás usando CP/M 2.2 deberías de cambiar de disco solamente cuando se te diga!

Published by Gilsoft International Ltd.,
2 Park Crescent, Barry, South Glamogan CF6 8HD
Telephone Barry (0446) 732765

P C W / C P C C P / M Art

La actualización de gráficos para CPC/PCW permite añadir gráficos basado en disco a cualquier aventura que hayas escrito que tenga 2.5k de TPA de espacio disponible. Nota que la actualización no es un paquete gráfico, asumimos que tu ya tienes uno. Acepta cualquier archivo de formato de pantalla que tengas. La pantalla funciona en Mode 1 (40 columnas, 4 colores) en el CPC por lo que los gráficos deben estar en este formato. También nota que los gráficos son diferentes para CPC y PCW por razones obvias!

Los archivo suministrados son:

CPC

MCPIC.COM	Crea un archivo PICCY en formato CPC
ADDARTC.COM	Añade o actualiza el programa gráfico a un juego PAW
ECPC.Z80	El programa gráfico
TICKET.MAP	Un ejemplo de archivo .MAP para el juego TICKET

PCW

MPPIC.COM	Crea un archivo PICCY en formato PCW
ADDARTP.COM	Añade o actualiza el programa gráfico a un juego PAW
EPCW.Z80	El programa gráfico
TICKET.MAP	Un ejemplo de archivo .MAP para el juego TICKET

Éstos archivos se encuentran en las respectivas caras del disco además de los archivos descritos en los manuales con PAW.

Vista General

El sistema está diseñado para ser lo más sencillo posible de utilizar. Puedes seleccionar para usar cualquier número de líneas de pantalla entre 2 a 20 para contener el gráfico – los archivos de gráfico contendrán sólo la cantidad de datos requeridos para ahorrar espacio en disco. El resto de la pantalla queda como ventana de texto. Tienes la opción de dejar una línea en blanco antes de la ventana de texto para separar gráficos y texto. Además el CLS en las descripciones puede desactivarse para proporcionar una ventana de scroll continuo.

Se parchea en el vector CLS de un juego en PAW. Digamos que añade un nuevo bit de programa al juego en PAW donde se limpia la pantalla! Este código extra funciona del siguiente modo:-

En el CPC la pantalla se en MODO 1 (40 columnas con 4 colores). Y bajo CP/M+ la línea de estatus se desactiva. La ventana de texto se ubica en la parte inferior de la pantalla, tras el número especificado de líneas para el gráfico y la posible línea en blanco.

Cuando PAW intenta un CLS, justo antes de describir una ubicación, el nuevo código toma el control. Si está selectado CLS en DESC el área de texto se limpiará. Nota que esto es llevado a cabo por el código de la interfaz en el CPC, los códigos CLS dados usando PAWINST no tienen significado.

Los colores especificados en la entrada del archivo .MAP para esta ubicación son establecidos en el CPC.

Si en el juego estamos a oscuras el área reservada para gráficos es borrada y se vuelve a PAW. Esto es en efecto una muestra del Gráfico 0, que es considerado como una pantalla en blanco.

CPC

MCPIC.COM	Crea un archivo PICCY en formato CPC
ADDARTC.COM	Añade o actualiza el programa gráfico a un juego PAW
ECPC.Z80	El programa gráfico
TICKET.MAP	Un ejemplo de archivo .MAP para el juego TICKET

Por otro lado, si el gráfico especificado en el archivo .MAP (ver abajo) para la ubicación actual no está ya en la pantalla, se carga desde disco. Si el gráfico no puede encontrarse se mostrará el mensaje adecuado en la pantalla de texto. Esto también sucede si ocurre un error mientras se carga el gráfico

Simple como 1.2.3.4.5 etc

Los pasos para generar un juego gráfico para CPC o PCW son:-

Escribir y probar tu juego usando PAW.
Grabar una versión compilada (.COM) usando PAWINT nombre C.
Usar una aplicación gráfica para dibujar algunos gráficos.
Convertir los gráficos a archivos PICCY usando MxPIC.COM
Escribir un archivo .MAP usando un editor de texto para describir el número de líneas, colores y qué ubicación contiene cada gráfico.
Usar el programa ADDARTx.COM para añadir el parche gráfico y los archivos .MAP al juego final.
Usar PAWINST para instalar el Intérprete para el número de líneas y columnas de texto.
Jugar el juego!

Hacer los archivos PICCY adecuados

MPIC archivo grafnum (.BIN) (lineas)
Crear un archivo desde un archivo .BIN en Modo 1 (uno que contenga cabecera AMSDOS de 128 bytes) llamado PICCYxxx.CPC (donde xxx = número de gráfico), conteniendo lineas 'líneas' (por defecto 12). El archivo .BIN se hace bajo AMSDOS:

```
MODE 1
SAVE “screen”,B,&C000,&4000
```

y es el formato para la mayoría de programas de dibujo disponibles.

Por ejemplo
MCPIC 2 falls
Creará el archivo PICCY002.CPC usando 12 líneas de FALLS.BIN

MCPIC 97 MYPIC.PIC 20
Creará el archivo PICCY097.CPC usando 20 líneas de MYPIC.PIC (nota que asumimos que el archivo contiene una cabecera AMSDOS de 128 bytes aunque la extensión sea distinta.)

MPPIC archivo grafnum (.PCP) (lineas)
Crear un archivo desde un archivo de captura de pantalla .PCP llamado PICCYxxx.PCW (donde

xxx = número de gráfico), conteniendo lineas 'líneas' (por defecto 12). El formato de los archivos .PCP es el utilizado por MasterScan/Paint.

Por ejemplo
MPPIC 2 falls
Creará el archivo PICCY002.PCW usando 12 líneas de FALLS.PCP

MPPIC 97 MYPIC.PIC 20
Creará el archivo PICCY097.PCW usando 20 líneas de MYPIC.PIC

Los Intérpretes parcheados buscarán los archivos en los formatos dados arriba. Nótese que si cambias cambias el número de lineas que utilizas para un gráfico en el archivo .MAP, tendrás que rehacer los gráficos. Ésto es porque si decides usar más líneas, habrá insuficiente datos en el archivo PICCY y te dará un error al cargarse. Y viceversa, si usas menos líneas tú tendrás demasiados datos en los archivos PICCY desperdiciando espacio de disco.

El archivo .MAP

En él se describe el formato de la pantalla y qué pantalla usar para cada ubicación. Puede prepararse del mismo modo que el archivo .SCE usando cualquier editor de texto.

Veamos un ejemplo imaginario para TEWK. Asumiremos que tienes tres gráficos, cada uno de 12 líneas:-

El Gráfico 1 será una pantalla de introducción (usado en ubicación 0).
El Gráfico 2 será dentro del barco (usado en las ubicaciones 1-3).
El Gráfico 3 será fuera del barco (usado en las ubicaciones 4-5).

El archivo .MAP sería:-

```
;MAP para gráficos de TEWK
;
;Líneas          Flags           Tinta 0         Tinta 1
12               1               0               26
;
;Los enlaces
;
;Ubi    Grá     Tin1    Tin2    Tin3
0       1       26      2       18              ;Puede ser cualquier color
1       2       26      3       4               ;Sólo porque sea el mismo piccy
2       2       26      43      14              ;no tiene por qué tener el mismo color
3       2       26      13      14              ;pero podría ser
4       3       26      4       5
5       3       26      4       5
;Listo
```

Cualquier linea en blanco será ignorada así como las que comiencen con un punto y coma ';' o un asterisco '*'.

La primera linea de datos válida debe contener el número de líneas usadas en cada gráfico. Los flags (ver abajo) y los valores iniciales para Tinta 0 (borde y fondo) y Tinta 1 (color del texto) en ese orden.

Los flags se definen como:-
0 – No hay línea en blanco para separar gráfico y texto. CLS en DESC
1 – Línea en blanco. CLS en DESC
2 – No hay línea en blanco. No hay CLS en DESC
3 – Línea en blanco. No hay CLS en DESC

Aunque le puedes dar otros valores – NO LO HAGAS. Están reservados para futuras expansiones y podrían tener efectos inesperados.

Las siguientes líneas entonces definen cualquier ubicación que contenga gráficos. En el orden; número de ubicación (0-251), número de gráfico a mostrar (0-250), Tintas 1 a 3 (0-26). Se usa una línea para cada ubicación pero éstas no necesitan estar en orden, y no hace falta especificar todas las ubicaciones.

Por ejemplo

```
11      1       26      0       0
10      1       26      0       0

8       2       26      0       0
52      2       26      0       0
```

definirá sólo las ubicaciones 8, 10, 11 y 52. Cualquier ubicación que no reciba una definición tendrá por defecto la Tinta 1 (sólo en CPC) especificada en la primera línea y gráfico 0. El gráfico 0 no existe realmente, es una pantalla en blanco. Se usa también para la oscuridad.

En el CPC el valor para la Tinta 1 puede cambiarse para cada ubicación si fuera necesario. Ésto debería tratarse con cuidado. Te sugerimos que intentes dibujar tus gráficos con Tintas 0 y 1 fijas (las que vayas a usar para el juego) y cambiar sólo los valores para las Tintas 2 y 3. Si necesitas cambiar la Tinta 1, haz un cambio pequeño cambiando a los tonos pastel o brillante que no son demasiado diferentes. Por ejemplo, amarillos y rojos o azules. Esto es porque el texto se escribe siempre con la Tinta 1 y puede distraer demasiado al jugador tener esos cambios de color continuamente. Esto es especialmente cierto si desactivas CLS en DESC para crear una ventana de scroll continuo ya que el texto de la pantalla cambiará de color!

Nota que los valores de TINTA serán ignorados por ADDART, así que puedes usar el mismo .MAP en CPC y PCW si quieres. Aunque parezca poco probable que se puedan dibujar exactamente los mismos gráficos en cada máquina.

Si defines una ubicación más de una vez la última definición tendrá prioridad, no se genera aviso...

Sólo puedes añadir el parche gráficos a un juego terminado. Uno que haya sido grabado con la opción 'C' en el Intérprete como un archivo .COM.

ADDARTx juego(.COM) (archivomapa(.MAP))
donde x = 'C' o 'P'.

Ésto realmente hace el trabajo de añadir los el programa/parche gráfico al juego e instala una copia legible por la máquina del archivo .MAP. El juego tomará por defecto la extensión .COM si omites el nombre de archivo de mapa que será por defecto el nombre del juego con una extensión .MAP. Si el juego ya contiene el parche gráfico el programa actualizará el código (en caso de nuevas versiones) y cambiará el archivo .MAP.

Debes usar la versión de ADDART correcta para la máquina en la que intentas hacer funcionar el programa – aunque todas las utilidades suministradas funcionarán bajo cualquier CP/M. Es sólo el programa final el que está limitado a una máquina! Si estás haciendo una versión tanto para PC como para PCW recuerda hacer una copia del archivo .COM. El mejor método sería tener dos archivos .COM por ejemplo TEWKC.COM y TEWP.COM o algo parecido.

Por ejemplo

ADDARTC TEWK

Añadirá los archivos TEWK.MAP y ECPC.Z80 al archivo TEWK.COM, cambiando el código necesario en TEWK.COM para introducir el programa gráfico en el vector CLS.

ADDARTP TICKET BLANK.TXT

Añadirá los archivos BLANK.TXT (asume que es un archivo .MAP) y EPCW.Z80 al archivo TICKET.COM

Debes instalar el archivo JUEGO.COM para cambiar el número de líneas y columnas que PAW cree que tiene la pantalla. Esto se consigue con el programa PAWINST del disco original de PAW.

Por ejemplo, en el CPC tienes 25 líneas. Si vas a usar las 12 líneas por gráfico por defecto y has establecido los flags para incluir una línea en blanco, quedarán 12 líneas para texto. Además como la pantalla ha cambiado a MODO 1 ahora sólo tenemos 40 columnas por línea y este cambio debe hacerse también.

En el PCW de las originales 32 líneas podrías tener 32-12-1=19. La pantalla permanece con 90 columnas así que no necesitas cambiar esa opción.

Si no cambias la instalación, el salto de línea y el 'More...” no funcionarán correctamente.

Como hemos mencionado antes, el CLS se lleva a cabo dentro del programa de gráficos del CPC. Así que la opción de instalación para éste es redundante en los programas que contengan gráficos.

Ejecutando el juego final

Una vez hayas llevado a cabo los pasos de arriba estarás listo para probar el programa con gráficos. Ésto es tan sencillo como escribir el nombre que el programa tenía antes.

NOTA: El parche de gráficos funcionará en una sola máquina. No intentes ejecutar un juego que tenga el parche gráfico de CPC en un PCW (o viceversa). Si tienes suerte se colgará...

Si aún no has colocado ningún archivo PICCY en tu disco obtendrás mensajes de 'Picture xxx not found'. Lo único que queda es añadir los gráficos al disco. De nuevo, has de colocar los gráficos correctos en el disco, archivos CPC para el CPC, etc...

Es posible que haya insuficiente memoria disponible para ejecutar el programa. En este caso se mostrará un mensaje de error y el juego se cerrará.

CP/M 2.2 y CP/M +

Un juego con el parche de gráficos instalado cuando se ejecute bajo CP/M+ puede reportar un error 'Insufficient common memory, remove RSX's'. Esto sucederá is demasiados programas residentes tales como monitores y RSX's están ocupando memoria por encima de C000h. La única solución es reiniciar el sistema CP/M sin ellos presentes.

Sólo CPC:

Todos los programas proporcionados funcionan bajo CP/M 2.2 o CP/M+. Además como el CLS se lleva a cabo ahora por el programa gráfico el mismo archivo .COM de tu juego funcionará bajo ambos sistemas operativos siempre que quepa en la TPA... El problema sucede para aquellos que intentan distribuir el juego final, y hecho así ocurre con los juegos sólo de texto. El TPA en CP/M+ es mucho mayor pero si quieres que tu juego corra en un 464 con DDI o en un 664 debes tener en cuenta que para estas máquinas sólo está CP/M 2.2 disponible.

Las opciones son las siguientes:

El juego debe escribirse para que quepa en un TPA de 43k. Como necesitas 2.5k para el parche gráfico eso significan 40k de TPA. Si compilas el juego dentro de este tamaño de TPA el Compilador dirá que se han conseguido añadir cerca de 2600 bytes libres para el parche gráfico. Si compilas en un TPA mayor aún puedes asegurarte de que quepa comprobando que la dirección final, dad por el Compilador tras la compilación, no exceda de 41472 - pero por favor, pruébalo en la TPA más pequeña antes de difundir el juego.

Proporciona dos versiones del archivo .COM conteniendo ambas el parche gráfico. Una para CP/M 2.2 y otra para CP/M+. Así, la versión para CP/M+ puede contener más texto, por ejemplo para la descripción de ubicaciones. Ésto es porque con 20k más de TPA puedes irte hasta 61440 y aún cabrá el parche gráfico al final del archivo! Los gráficos serán iguales para ambos archivos .COM.

Nota que debido a la necesidad de desactivar el emulador de terminal de CP/M+ en CPC, y por tanto de la linea de estatus, el mensaje de error 'Disc not ready' no se producirá. En su lugar el parche gráfico reportará 'File not found' si no hay disco en la unidad. Errores de hardware serios podrían tener resultados inesperados... CPM 2.2 producirá los mensajes 'Retry, Ignore, Cancel?' en la ventana de texto actual , como es de esperar.

- - - - - - - - - - - -

Distribuido por:
GILSOFT, 2Parl Crescent, Barry, South Glamorgan. CF6 8HD