DAAD Ready! v0.6 BETA

Este documento usa partes del manual de DAAD v2 (C) Infinite Imaginations 1988/89/91, que fue escrito originalmente por Tim Gilberts, organizado de nuevo en formato actual por Pedro Fernández, partiendo de lo encontrado en los discos de Andrés Samudio, y tiene correcciones realizadas por la experiencia de muchos usuarios, con especial mención a Stefan Vogt. Este manual ha sido creado, añadiendo nuevas secciones, revisando otras, y retirando aquellas que no son relevantes para DAAD Ready, por Uto.

POR FAVOR TEN EN ECUENTA QUE ESTE NO ES EL MANUAL DE DAAD, SINO UNA ADAPTACIÓN HECHA PARA DAAD READY

LEE ESTE MANUAL PARA SABER COMO USAR DAAD READY, PROBABLEMENTE NO NECESITARAS LEER EL MANUAL ORIGINAL, PERO UNA VEZ QUE CONOZCAS BIEN DAAD READY, PUEDES LEERLO.

TABLA DE CONTENIDOS

Introducción

DAAD Ready hace fácil a desarrolladores tener sus aventuras funcionando en emuladores sin tener que preocuparse por construir discos, transferir ficheros, y otras cosas que requerían y requieren mucho esfuerzo en el mundo actual de emuladores, y en el de las máquinas originales.

Con *DAAD Ready* solo tendrás que concentrarte en crear tu juego, porque construir los ficheros .DSK, .D64, etc. es tan fácil como hacer doble clic en el fichero .BAT de su plataforma (por ejemplo, C64.BAT para el fichero .D64 de Commodore 64).

Debido a esto, en este manual encontraréis referencias al DAAD, la herramienta original, y a *DAAD Ready,* el paquete que tenéis entre manos, que incluye DRC (el nuevo compilador, por Uto), los intérpretes originales de DAAD (por Infinite Imaginations), el nuevo intérprete de MSX2 (por NatyPC), el nuevo intérprete para Plus/4 (por Imre Szell basado en el original de C64), el nuevo intérprete de PC (por Uto), la extensión Maluva para usar gráficos basados en disco en lugar de los antiguos vectoriales, por Uto, y muchas utilidades para empaquetar y probar los juegos (emuladores y otras herramientas). El uso de todo esto es transparente al programador, que solo tendrá que hacer doble clic sobre el BAT correspondiente.

El principio básico tras DAAD es facilitar un Sistema en el que una aventura se escribe una vez, y después funciona en todas las máquinas. Por supuesto solo es realmente posible sacrificando algunas funcionalidades. Por tanto, se adquiere un compromiso y una cierta cantidad de trabajo extra para poder hacer que cada máquina llegue a su capacidad. El núcleo del juego permanece igual facilitando así las pruebas de la aventura.

El principio básico de DAAD Ready es hacer todo eso posible en un simple click, y utilizar medios de almacenamiento modernos (razón por la cual no se usan gráficos vectoriales).

DAAD Ready puede crear juegos para:

- **PC/DOS** (VGA 16 colores)
- **Sinclair Spectrum 48k** (cinta, sin gráficos)
- **Sinclair ZX Spectrum Next** (tarjeta SD)
- **ZX-Uno** (tarjeta SD)
- **ZX Spectrum +3** (disquete).
- **Amstrad CPC** (disquete, para 664 or 6128). También para la interfaz M4.
- **Commodore 64 o 128** (disquete)
- **Commodore Plus/4** (disquete)
- **MSX 1** con al menos 64K de RAM (disquete)
- **MSX 2** (disquete)

- **Amstrad PCW** 8000/9000 Series (sin gráficos)
- **Atari ST (sin gráficos)**

Ten en cuenta que DAAD Ready no es capaz de generar aventuras con gráficos para Atari ST y Amstrad PCW. Esto es así porque se desconoce el formato en el que se almacenan los gráficos en estos ordenadores, y por tanto la única manera de añadirlos es usar las herramientas originales que venían incluidas en DAAD, que son para DOS y no tienen versión de línea de comando.

Además, DAAD Ready no genera ficheros para Amiga, a pesar de que DAAD sí puede. En este caso, se debe fundamentalmente a la imposibilidad de incluir un emulador funcional dentro del paquete DAAD Ready, dado que un emulador de Amiga requiere de ROMs cuya propiedad intelectual no ha sido liberada.

DAAD usa un lenguaje de programación creado especialmente para escribir aventuras. Esta vagamente basado en el de QUILL y PAW de Gilsoft, y si conoces dichos sistemas será útil. Hay una sección más adelante que indica las diferencias con PAW.

DAAD Ready puede ser dividido en 4 áreas funcionales:

El fichero fuente

Las fuentes (ficheros. DSF, DAAD Source File) son una colección de tablas que contienen toda la información para definir una Aventura. Esto incluye la descripción de localidades, peso y descripción de objetos, y la lógica del juego.

Los ficheros DSF tienen el formato de un fichero de texto ASCII, usando la codificación ISO-8859-1. Por favor ten en cuenta que muchos editores modernos tienden a usar codificación UTF-8 por defecto, lo cual puede tener un efecto colateral si estás creando un juego que no esté en inglés con caracteres especiales de ese idioma (“ñ”, “á”, “è”, etc.). Para juego en inglés eso no importa. Asegúrate de que tu editor está editando tus fuentes en ISO-8859-1.

El Compilador

El compilador funciona en la máquina de desarrollo solamente. El mismo comprueba el fichero fuente en busca de errores, y lo convierte en un fichero de base de datos específico para la máquina de destino (.DDB). Se trata de un fichero que los intérpretes pueden entender. *DAAD Ready* incluye el compilador DRC, un nuevo compilador hecho por Uto a finales de la década de 2010.

En realidad, no es necesario que sepas demasiado sobre el compilador,

porque *DAAD Ready* se encarga de llamarlo por ti, pero sí podrías encontrar que tu juego no compila bien porque has cometido algún error al modificar el fichero fuente. Si así es, verás el error en pantalla al ejecutar el .BAT correspondiente. Arregla el error, o si no sabes de donde viene, deshaz los últimos cambios e inténtalo de nuevo.

Los intérpretes

Son el verdadero corazón de DAAD. Hay un intérprete para cada máquina soportada por DAAD, que ejecutan el juego usando una serie de rutinas que llevan a cabo las tareas que necesita el juego de aventuras.

Facilitan la parte que es machine-independent en DAAD. *DAAD Ready* también se encarga de los intérpretes, creando un fichero DSK o D64, en el que incluye el correspondiente sin que tú tengas que pensar cual pones.

Los Gráficos

A diferencia del DAAD original, DAAD Ready no usa gráficos vectoriales para las imágenes de localidad, en su lugar utiliza gráficos bitmap. Para añadir gráficos, *DAAD Ready* usa la extensión Maluva (por Uto). Facilitaremos más detalle sobre esto más tarde en este manual.

Ayuda

Si necesitas más ayuda entendiendo cualquier asunto sobre DAAD, puedes encontrar ayuda en los siguientes grupos de Telegram:

https://t.me/RetroAventuras (español)
https://t.me/Advent8bit (inglés)

Idiomas soportados

A pesar de que DAAD solo soportaba español e inglés originalmente, se ha añadido un soporte limitado para portugués y alemán. Puedes saltar esta sección si vas a crear un juego en inglés o español.

"Limitado" significa que aunque será capaz de crear juegos que muestren textos usando caracteres de dichos idiomas como "ß" o "õ", el intérprete que subyace será siempre el inglés (para alemán) o el español (para portugués), por lo que el jugador no será capaz de dar órdenes al juego que incluyan esos caracteres, y el *parseado* de las mismas, pensará que es inglés o español, con las consecuencias que eso pueda tener. Además, afecta al listado de objetos:

- Si un objeto comienza por "a" o "some" en juegos en inglés, cuando es añadido a un mensaje como "You take the _." Ese guion bajo es sustituido por el texto del objeto sin el artículo. Así, un objeto llamado "a lamp" mostrará "You take the lamp." Cuando lo coges, en lugar de "You take the a lamp."
- Si la descripción de un objeto empieza por "una", "un", "una", "unos" en un juego en español, es artículo definido es cambiado por el indefinido, así que "Tomas _." funcionará, y "una lámpara" generará "Tomas la lámpara." Al ser cogida.

Lamentablamente, este funcionamiento es muy específico del lenguaje, así que los juegos en alemán o portugués no tendrán sus artículos modificados, porque no coinciden con lo esperado (además, el alemán tiene tres géneros). En consecuencia, una solución razonable ha sido tomada para esos idiomas: en portugués, los mensajes usarán el artículo indefinido, y en alemán se evita usar el nombre del objeto en la respuesta y contesta con un simple "OK, cogido."

Otros lenguajes pueden ser usados con DAAD Ready de una manera similar al portugués y alemán. Por favor únete a los grupos de Telegram si quieres crear soporte para otros lenguajes. Estos son los caracteres que soporta DAAD Ready aparte de los del inglés, si cuadran con tu idioma en general será posible:

ª ¡ ¿ á é í ó ú ñ Ñ ç Ç ü Ü à ã â ä è ê ë ì î ï ò ô ö õ ù û Ý
Á É Í Ó Ú Â Ê Î Ô Û À È Ì Ò Ù Ä Ë Ï Ö Ü ý Ý þ Þ å Å ð Ð ø Ø ß

Gráficos

Como se ha indicado anteriormente, *DAAD Ready* usa gráficos bitmap para las aventuras, en lugar de gráficos vectoriales, y utiliza la extensión Maluva para ello.

Para añadir gráficos, necesitarás ficheros diferentes para cada plataforma, dado que las plataformas y sus modos gráficos son muy diferentes.

DAAD Ready solo maneja gráficos para las plataformas de 8-bit, a excepción de PCW, y para PC. Para añadir gráficos a PCW, Amiga o Atari ST, por favor mira el manual original del DAAD

Para añadir gráficos para tus aventuras, debes colocar las imágenes en la carpeta IMAGES de DAAD Ready, excepto para CPC, que debes ponerlas en la carpeta CPC que está dentro de IMAGES, y para PC, que debes ponerlos en la carpeta PC que hay dentro de IMAGES.

Todas las imágenes que añadas deberán ser a pantalla completa para la máquina respectiva, es decir 256x192 para Spectrum o MSX1, 160x200

para C64 (HiColor), 320x200 para Plus/4, etc. A pesar de ello *DAAD Ready* solo va a usar las 96 primeras líneas de pixeles de cada gráfico que pongas, por lo que es indiferente lo que haya en esos gráficos a pantalla completa por debajo de la línea 96. Hay una excepción a esto: los gráficos para PC deben ser ya de 96x200 pixeles.

Para añadir gráficos, añade ficheros en las carpetas mencionadas, en el formato correspondiente:

- Spectrum SCR para Spectrum +3 o ESXDOS
- PCX indexados de 256 colores para Spectrum Next
- MLT o MLT+PAL para ZX-Uno
- La versión cinta de Spectrum no soporta gráficos en DAAD Ready
- SCR + PAL para Amstrad CPC, modo 0
- Koala (KLA) para C64 HiColor o ART para HiRes
- PRG para Commodore Plus/4
- SC2 para MSX 1
- SC8 para MSX 2
- 256 color indexed PCX file for PC

Todos estos formatos son muy comunes y fáciles de crear para cada Plataforma.

Para que DAAD Ready entienda los gráficos, estos deben ser nombrados siguiendo la pauta de que funcionarán para cada localidad si coinciden en número usando tres dígitos. Así, si quieres un gráfico para la localidad 12, deberás poner el gráfico 012, con la extensión que corresponda para la máquina en cuestión (012.KLA, o 012.PRG, o 012.SCR, etc.)

También puedes añadir gráficos no relacionados con localidades, simplemente metiendo un gráfico de una localidad que no exista. Por ejemplo, si pones el 200.KLA este será añadido, y luego lo puedes mostrar usando una llamada XPICTURE (ver los condActos Maluva CondActos más adelante).

Ten en cuenta que *DAAD Ready* es una herramienta sencilla para ayudarte a empezar a trabajar con DAAD y no preocuparte de muchas cosas. Hacer juegos con más o menos de 96 líneas es posible, hacer gráficos vectoriales es posible, etc. pero quedan fuera del ámbito para el que DAAD Ready se ha creado.

Herramientas recomendadas

Muchos de los formatos de imágenes pueden ser generados fácilmente con emuladores, esta es una lista de herramientas recomendadas (para Windows):

Spectrum .SCR 256x192	Es exportado por casi cualquier emulador. ZX Paintbrush también

	lo maneja.
Spectrum Next .PCX 256x192	Fácil de generar con Gimp. Simplemente asegúrate de que la paleta está como indexada, y que tu imagen tiene 17 colores o más, o Gimp generará un PCX de 16 colores que no vale.
ZX-Uno MLT/PAL 256x192	Los ficheros MLT pueden crearse con ZX-Paintbrush o Image2Ulaplus. Si el MLT mide 12,288 bytes, la información de paleta no está incluida, por lo que hay que exportar también esa información en un TAP (ZX-Paintbrush puede exportarla paleta así).
CPC .SCR and .PAL 320x200	CPCImgConv es una herramienta fantástica para conseguir las imágenes. Tener en cuenta que el color 0 de la paleta se usará para el fondo del texto, y el 1 para el texto en sí. Así que o conseguís que el 0 sea negro y el 1 blanco, o al menos que esos colores tengan un buen contraste. Exporta tanto el SCR como el fichero .PAL para cada localidad.
C64 KLA (160x200) o ART (320x200)	Multipaint es una excelente herramienta para manejar imágenes C64 Multicolor.
Plus/4 320x200	Las imágenes pueden ser creadas con la aplicación Boticceli HiRes y exportadas como 001.PRG, 002.PRG, 003.PRG, etc. Botticeli es una aplicación para Plus/4.
MSX 1 .SC2 256x192	Hay un conversor bastante bueno en https://msx.jannone.org/conv/. Asegúrate que usas la paleta estándar MSX 1.
MSX 2 .SC8 256x212	Mismo conversor que MSX1 pero usa SC8 y paleta MSX2.
PC	Fácil de generar con Gimp. Simplemente asegúrate de que la paleta está como indexada, y que tu imagen tiene 17 colores o más, o Gimp generará un PCX de 16 colores que no vale.

Ten en cuenta que los intérpretes originales de DAAD usaban el modo HiRes para C64 y el modo 1 para CPC. Esos no son los mejores modos para C64y CPC si piensas en los gráficos, pero son los modos que soportan mejor pintar gráficos y pintar texto a la vez. La extensión

Maluva que usa DAAD Ready permite usar lo que se llama un "split mode" (modo partido), en el que la parte superior de la pantalla usa un modo gráfico y la inferior otro. Esto permite juegos de C64 con la parte superior en HiColor y la inferior en HiRes, y juegos en CPC con la parte superior en modo 0 y la inferior en modo 1.

Lamentablamente, mantener esos modos partidos funcionando requieren complicadas rutinas de interrupción que son difíciles de mantener cuando se lee de discos. Esto es por lo que verás que, en estos ordenadores, la imagen o la pantalla completa son ocultadas mientras lees de disco a siguiente imagen, y que incluso puedes ver algunos parpadeos y efectos extraños cuando se leen mensajes con XMES o XMESSAGE (ver condActos Maluva más adelante)

Si no quieres que pase eso, siempre puedes usar los modos tradicionales, es decir HiRes y Modo 1 para toda la pantalla en C64 y CPC respectivamente. Para hacer eso, edita el fichero CONFIG.BAT y cambia la línea que pone SET SPLITSCR=SPLIT a SET SPLITSCR=NOSPLIT. A partir de ahí asegúrate de que usas las imágenes adecuadas (SCR en Modo 1 para CPC, o ficheros ART en lugar de KLA para C64).

Cómo usar DAAD Ready

Este es un resumen de los puntos principales para usar DAAD Ready para generar un juego:

Para empezar, ejecuta cualquiera de los .BAT (ZXPLUS3.BAT por ejemplo). Te preguntará qué idioma va a usar el juego y un juego con una plantilla vacía será compilado y ejecutado. Limítate por ahora a cerrar el emulador cuando este se abra.

Bien, una vez que has ejecutado cualquiera de los .BAT tendrás un fichero TEST.DSF en la carpeta de *DAAD Ready* Ese fichero es el que debes editar (recuerda, editor en (ISO 8859-1) si el juego que haces no está en inglés.

Puedes hacer cambios sabiendo como programar para DAAD, lo que se explicará más adelante en este manual.

Cada .BAT compila juegos para la máquina de destino correspondiente, ten en cuenta que habrá una pausa antes de lanzar el emulador, en la que puedes pulsar Ctrl + C si no quieres ejecutarlo.

También ten en cuenta que a veces antes de lanzar el emulador te salen en pantalla unas notas sobre cosas que debes hacer cuando el emulador se lance, porque no todos los emuladores permiten cargar el juego directamente, a veces tienes que hacer a mano el típico LOAD o RUN.

Finalmente, ten en cuenta que cuando compilas el juego, se creará un fichero DSK o D64 o lo que corresponda a esa máquina en la carpeta RELEASE de DAAD Ready

Otros detalles

- Una vez seleccionado un idioma, se considerará que estás haciendo un juego en ese idioma. Si quieres crear uno en otro lenguaje, por ahora solo se puede instalar otra copia de DAAD Ready en otra carpeta. Es matar moscas a cañonazos, pero ahora mismo no hay otra opción
- Visual Studio Code es el editor recomendado, porque tiene un addon de Chris Ainsley que permite que el Código de DAAD se coloree, lo que ayuda mucho a ver las cosas.

Programando en DAAD

Existe un tutorial hecho por Uto (en inglés) en la siguiente URL: https://medium.com/@uto_dev/a-daad-tutorial-for-beginners-1-b2568ec4df05

Lo que sigue es una guía más técnica que ese tutorial, pero necesaria para entender DAAD en profundidad.

Vista general

Cuando un intérprete de juego se inicia, empezará a ejecutar la información que pusiste en el fichero TEST.DSF. Si le echas un vistazo al fichero DSF, verás varias secciones para definir objetos, localidades, etc. y al final los procesos. Los procesos son el código de programación de DAAD, que determina la lógica del juego. Cuando una aventura comienza, el proceso 0 es el primero en ejecutarse (sección "/PRO 0").

Ten en cuenta que el fichero TEST.DSF que viene con DAAD Ready ya contiene código que hace que DAAD se comporte muy parecido a PAW, por lo que si conoces PAW (o Superglús, SINTAC, ngPAWS, NMP, SKC) te será fácil adaptarte.

Los flags

Los flags son como contenedores que pueden almacenar un valor. Puedes cambiar ese valor y comprobar el valor más tarde, y así saber si algo en concreto pasó anteriormente. Por ejemplo, puedes dar a un flag el valor 1 cuando un malvado troll ha huido debido a la acción del jugador, y después comprobar si ese flag contiene el valor 1 cuando el jugador trata de cruzar el puente donde estaba el troll, de modo que, si vale 1, le dejamos cruzar, y si no, le decimos que el troll está allí y que no puede pasar.

Hay 256 flags disponibles en DAAD, así que tienes más que suficiente para tus puzles, a pesar de que algunos de ellos tienen un funcionamiento interno por lo que no los podrás usar. Por ejemplo, el flag 38, también conocido como "fPlayer", contiene el número de la localidad donde está el jugador, así que, si lo cambias, en realidad estarás llevando al jugador a otro lugar. Puedes usarlo también para saber si el jugador está en determinada localidad (ej: permitirle dormir en el dormitorio, pero no en el baño).

Los flags del sistema, como se les llama, están explicados más tarde en esta sección.

Inicialización

Cuando el juego se inicia, el intérprete realiza la inicialización solo una vez, la cual implica que todos los flags se ponen al valor 0 y la pantalla es borrada. Nótese que es por esto por lo que el jugador siempre empieza en la localidad 0, porque el flag 38 (fPlayer) vale cero.

Inicio

El intérprete llama al proceso 0, cuando sale de ese proceso se llama de nuevo al sistema operativo o se reinicia el ordenador. Normalmente esto solo ocurrirá si el jugador realizar un fin de partida y dice que no quiere jugar de nuevo.

Ejecución del proceso 0

El proceso 0 normalmente contiene el bucle principal. En realidad, es algo que ya viene en el TEST.DSF y está preparado para funciona como en PAW, pero explicaremos como se procesa un proceso (Valga la redundancia):

La figura 1 y la siguiente sección describen el proceso de escaneado de una tabla de procesos, que aplica al proceso 0 y a cualquier otro. La única diferencia del proceso 0 es que, como decíamos, cuando termina, se vuelve al SO.

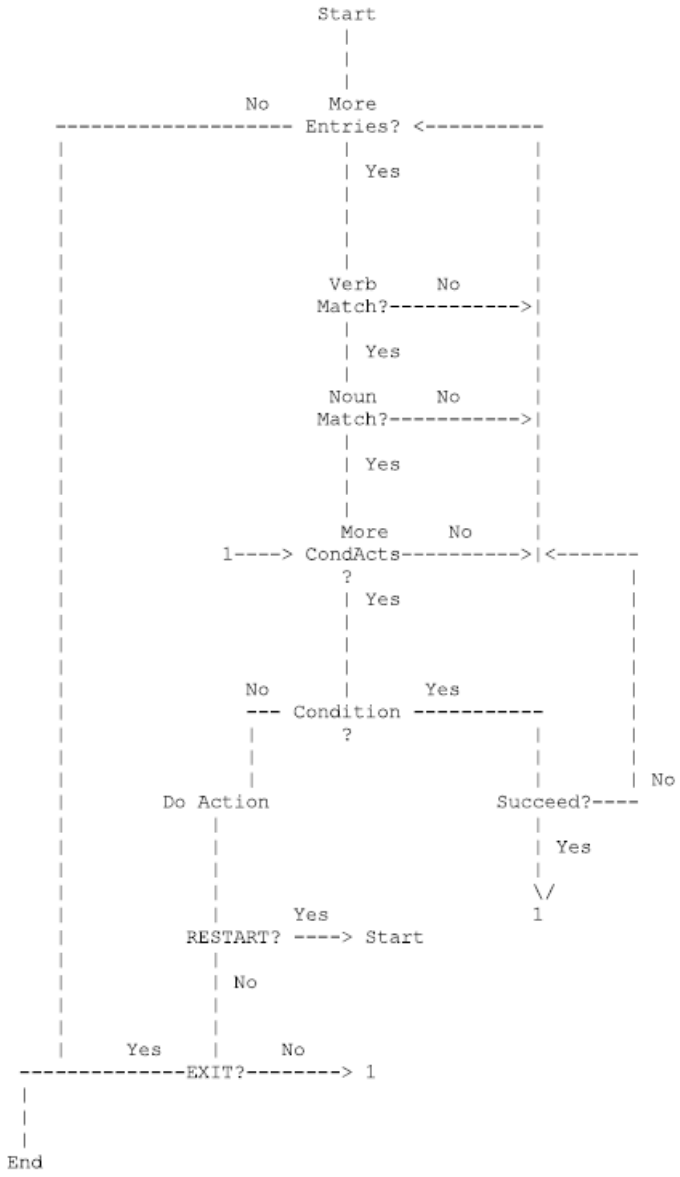

Figura 1

Ejecución de un proceso

Básicamente DAAD buscará en cada entrada (comienzan por el carácter ">") hasta que el proceso se le acabe. Asumiendo que hay alguna entrada, comprobará que el verbo y nombre de estas correspondan con el verbo y nombre de la orden dada por el jugador, al cual habrá sido extraída por una orden PARSE en el proceso.

El uso de la palabra "_" en lugar del verbo o el nombre, implica que esa entrada no realizará comprobación del verbo o el nombre. Así una entrada del tipo "> _ _" se ejecuta con indiferencia de lo que haya escrito el jugador. Una entrada "> ABRIR _" se ejecutará si el jugador escribió "ABRIR" o un sinónimo, independientemente de qué pidiera abrir (ABRIR PUERTA, ABRIR CAJA, etc.), y una entrada "> _ LLAVE" se ejecutará si el jugador pide hacer cualquier cosa con la llave (MIRAR LLAVE, DEJAR LLAVE, COGER LLAVE, etc.)

Una vez que vemos que una entrada cumple las condiciones verbo/nombre, ejecutaremos los condActos de la entrada uno a uno. Cuando un condActo es de tipo condición, si tiene éxito, se ejecutará el siguiente, y si no nos saldremos de esa entrada y no ejecutaremos más condActos. Si se trata de una acción, esta será ejecutada.

Las acciones pueden ser divididas en cinco grupos según su funcionamiento:

- RESTART, que Vuelve la ejecución al principio del proceso 0, saliendo de cualquier proceso en el que estuviera. Para los programadores de PAW: equivale al antiguo DESC.

- END/EXIT, que finalizan el juego y saltan a reiniciarlo o al sistema operativo.

- Salida: cualquier acción que detenga la ejecución del proceso actual, como por ejemplo DONE.

- Salida condicional: cualquier acción que pare la ejecución del proceso actual en determinadas condiciones, como GET, PUTIN, etc.

- Normal: cualquier acción que haga su función y nada más, como COPYFF, PLUS, etc.

Nota: varios condActos de DAAD funcionan diferente a como eran en PAW, así que aseguraos de mirar bien cómo funcionan si usasteis PAW.

QUIT es un ejemplo, porque en DAAD es realmente una condición. El uso de DESC puede ser bastante confuso, porque en DAAD es mucho más parecido a un "MESSAGE" pero para texto de localidad, mientras que la función que hacía DESC en PAW, realmente la hace RESTART en DAAD.

CondActos

Aquí sigue una descripción detallada de cada condActo que puede ser incluido en una entrada. Se dividen en grupos dependiendo del asunto que tratan en DAAD, como flags, objetos, etc. y da algunas pistas sobre su posible uso en ocasiones.

Para aquellos acostumbrados a PAW: tened cuidado. Algunas acciones han desaparecido, otras han cambiado su función. Aseguraos de comprobar la función en DAAD. Por ejemplo, TURNS y SCORE se han borrado, y ahora deben ser programadas por código (con PRINT y DPRINT, por ejemplo). También TIMEOUT, PROMOT y GRAPHIC se han ido, dado que "HASAT fTimeout" o "LET fPrompt x" y "LET fGFlags expresión" harán lo mismo.

Varias abreviaturas serán usadas:

locno. es un número de localidad válido

locno+ es un número de localidad válido, o 252 (no creado), 253 (puesto), 254 (llevado) o 255 (HERE, aquí, la localidad actual del jugador).

mesno. es un número de mensaje válido

sysno. es un número de mensaje del sistema válido

flagno. es cualquier flag (0 a 255).

procno. Es un número de proceso válido

word es una palabra del tipo requerido, que está presente en el vocabulario, o un guion bajo "_" que se asegura que no hay ninguna palabra de ese tipo en la orden.

value es un valor de 0 a 255.

Indirección

El primer parámetro de la mayoría de las acciones o condiciones puede usar indirección. Esto se indica poniendo una arroba (@) delante del primer parámetro. Hacerlo así causará que el valor de dicho parámetro no sea considerado el valor definitivo, sino un número de flag donde buscar el valor definitivo. Solo el primer parámetro puede usar indirección, pero esto provee de una funcionalidad bastante potente.

Ej. 'MESSAGE @100' escribirá no el mensaje 100, sino el mensaje cuyo número esté en el flag 100.

Condiciones

Condiciones sobre la localidad del jugador

AT locno.

Tiene éxito si la localidad actual es locno.

NOTAT locno.

Tiene éxito si la localidad actual no es locno.

ATGT locno.

Tiene éxito si el número de la localidad actual es mayor que locno.

ATLT locno.

Tiene éxito si el número de la localidad actual es menor que locno.

Condiciones sobre la localidad actual de un objeto

PRESENT objno.

Tiene éxito si el objeto objno. lo llevamos, lo llevamos puesto o está en la localidad actual del jugador.

ABSENT objno.

Tiene éxito si el objeto objno. no lo llevamos, no lo llevamos puesto u no está en la localidad actual del jugador.

WORN objno.

Tiene éxito si el objeto objno. lo llevamos puesto.

NOTWORN objno.

Tiene éxito si el objeto objno. no lo llevamos puesto.

CARRIED objno.

Tiene éxito si el objeto objno. lo llevamos.

NOTCARR objno.

Tiene éxito si el objeto objno. no lo llevamos.

ISAT objno. locno+

Tiene éxito si el objeto objno. está en la localidad locno.

ISNOTAT objno. locno+

Tiene éxito si el objeto objno. no está en la localidad locno.

Condiciones que comparan valores de flags

ZERO flagno.

Tiene éxito si Flag flagno. vale cero.

NOTZERO flagno.

Tiene éxito si Flag flagno. no vale cero.

EQ flagno. value

Tiene éxito si Flag flagno. vale el valor "value".

NOTEQ flagno. value

Tiene éxito si Flag flagno. no vale el valor "value".

GT flagno. value

Tiene éxito si Flag flagno. vale un valor superior a "value".

LT flagno. value

Tiene éxito si Flag flagno. vale un valor inferior a "value".

SAME flagno 1 flagno 2

Tiene éxito si ambos flags valen lo mismo.

NOTSAME flagno 1 flagno 2

Tiene éxito si ambos flags no valen lo mismo.

BIGGER flagno 1 flagno 2

Tiene éxito si el flag flagno1 vale más que el flag flagno2.

SMALLER flagno 1 flagno 2

Lo contrario del anterior. En realidad, no hace falta porque cambiando el orden de los parámetros del anterior valdría, pero hace el programa más legible.

Condiciones de comprobar otras palabras en la orden del jugador

En general es mejor usar estas condiciones con una palabra específica, o con la ausencia de esta usando "_", si la presencia o ausencia de esta es requerida para una situación concreta. Hacerlo de manera genérica solo reduce la flexibilidad del juego y puede llevar a la frustración del jugador. Es decir, si fuerzas al jugador a escribir "ABRIR CANDADO CON LLAVE" en lugar de "ABRIR CANDADO", cuando el jugador lleva una llave, es añadir dificultad a una acción bastante natural. Fuerza el "CON LLAVE" solo cuando poner CON LLAVE no sería lo más natural.

ADJECT1 word

Tiene éxito si el adjetivo del primer nombre de la orden del jugador es el especificado, o bien si no hay un adjetivo y el parámetro era

"_".

ADVERB word

Tiene éxito si el adverbio de la orden del jugador es el especificado, o bien si no hay un adverbio y el parámetro era "_".

PREP word

Tiene éxito si la preposición de la orden del jugador es el especificado, o bien si no hay una preposición y el parámetro era "_".

NOUN2 word

Tiene éxito si el segundo nombre de la orden del jugador es el especificado, o bien si no hay un segundo nombre y el parámetro era "_".

ADJECT2 word

Tiene éxito si el adjetivo del segundo nombre de la orden del jugador es el especificado, o bien si no hay un segundo adjetivo y el parámetro era "_".

Condiciones para asuntos aleatorios

Puedes facilitar unas posibilidades aleatorias de que un jugador se caiga de un árbol, reciba un rayo, o de que un Puente se hunda. No abuses de esto o dale al jugador una manera de evitar el problema, como botas de goma para el rayo o similar.

CHANCE porcentaje

Tiene éxito si porcentaje es menor o igual que un número aleatorio de 1 a 100 (inclusivos). así, CHANCE 50 tiene una posibilidad de éxito del 50%.

Condiciones con un resultado VERDAD/FLASO en llamadas a procesos

ISDONE

Tiene éxito si la última tabla de procesos ejecutada realizó al menos una acción. Esto es útil para comprobar el resultado de éxito o fallo de un proceso. Una acción DONE causará el estado de "hecho", así como cualquier condActo de acción ejecutado al acabar la tabla, siempre que esa acción no sea NOTDONE.

Ver también ISNDONE y NOTDONE.

ISNDONE

Tiene éxito si el último proceso ejecutado terminó sin ninguna acción ejecutada, o con una acción NOTDONE.

Condiciones para los atributos de objeto

HASAT/HASNAT value

Comprueba que el atributo de objeto especificado esté activo (HASTAT) o inactivo (HASNA). Los valores 0-15 son para los 16 atributos de objeto. Además, hay determinados valores que comprueba otras cosas en los flags del sistema de DAAD:

Símbolo	Flag	Número de Atr.	Descripción
WEARABLE	57-Bit7	23	El objeto actual es prenda
CONTAINER	56-Bit7	31	El objeto actual es contenedor
LISTED	53-Bit7	55	Si los objetos son listados por LISTOBJ
TIMEOUT	49-Bit7	87	Si hubo un timeout en el último input
GMODE	29-Bit7	257	Gráficos disponibles (sin mucho sentido en DAAD Ready)
MOUSE	29-Bit0	240	Ratón disponible

Estas opciones pueden ser comprobadas/activadas con los valores dados en TEST.DSF.

Por ejemplo, la opción de pintar escondido y mostrar la paleta al final del CPC puede hacer así:

```
HASNAT GA_MDRW; Si no temenos magic draw
PLUS fGFlags GO_MDRW
```

Nota: los símbolos en el formato GA_xxx son valores para comprobar con HASAT/HASNAT, mientras que los valores GO_xxx son usados para cambiar el valor a 0 o 1, usando PLUS/MINUS del bit.

Como otro ejemplo, la condición TIMEOUT de PAW es implementada en DAAD como:

```
HASAT TIMEOUT
```

Esto también permite que una acción 'NOTTIMEOUT' sea implementada usando HASNAT.

Puedes por supuesto asignar tus propios valores a partes de un flag

usando HASAT y HASNAT, son sistemas genéricos de control de los bits para los primeros 64 flags.

Condiciones para interactuar con el jugador

INKEY

Es una condición que tendrá éxito si el jugador está pulsando una tecla cuando se ejecute. En las máquinas de 16 bit, los flags fKey1 y fKey2 (60 y 61) se actualizarán con los valores correspondientes a la tecla en el IBM estándar ASCII code. En 8 bit solo fKey será modificado, y su valor es diferente para cada máquina.

QUIT

Se muestra el mensaje del Sistema 12 ("¿Estás seguro?") y se llama a la rutina de input. La condición tendrá éxito si el jugador responde con una frase que empieza por la letra que hay en el mensaje del sistema 30 (“S”). En caso contrario, el resto de lo tecleado por el jugador es descartado y saltamos a la siguiente entrada.

Acciones

Acciones que manipulan la posición de los objetos

GET objno.

Si el objeto objno lo llevamos o lo llevamos puesto, se imprime el mensaje del Sistema 25 ("Ya tengo _.") y una acción NEWTEXT y DONE son ejecutados.

Si el objeto objno no está en la localidad actual, se imprime el mensaje del Sistema 26 ("No veo eso por aquí.") y una acción NEWTEXT y DONE son ejecutados.

Si el peso total de los objetos que lleva el jugador, más las prendas que lleva puestas, más el peso del objeto en cuestión excede el máximo peso que puede llevar el jugador (flag 52) se imprime el mensaje del Sistema 43 ("@ pesa demasiado.") y una acción NEWTEXT y DONE son ejecutados.

Si el número de objetos que lleva el jugador (flag 1) es mayor o igual que el máximo de objetos que puede llevar (Flag 37), se imprime el mensaje del Sistema 27 ("Llevo demasiadas cosas.") y una acción NEWTEXT y DONE son ejecutados.

Si no ocurre nada de lo anterior la posición del objeto en cuestión es cambiada a "llevado" (localidad 254), el flag 1 es incrementado en una unidad, y se imprime el mensaje del sistema 36 ("Ahora llevo _.").

DROP objno.

Si el objeto objno es una prenda que el jugador lleva puesta se imprime el mensaje del Sistema 24 ("No puedo. Llevo puesto _.") y una acción NEWTEXT y DONE son ejecutados.

Si el objeto objno está en la localidad actual, pero ni llevado ni puesto, se imprime el mensaje del Sistema 49 ("No tengo _.") y una acción NEWTEXT y DONE son ejecutados.

Si el objeto objno no está en la localidad actual, se imprime el mensaje del Sistema 28 ("No tengo eso.") y una acción NEWTEXT y DONE son ejecutados.

Si no ocurre nada de lo anterior, el objeto es movido a la localidad actual, el flag 1 es decrementado en una unidad, y se imprime el mensaje del sistema 39 ("He dejado _.").

WEAR objno.

Si el objeto objno está en la localidad actual, pero ni llevado ni puesto, se imprime el mensaje del Sistema 49 ("No tengo _.") y una acción NEWTEXT y DONE son ejecutados.

Si el objeto objno es una prenda que ya llevamos puesta, se imprime el mensaje del Sistema 29 ("Ya llevo puesto _.") y una acción NEWTEXT y DONE son ejecutados.

Si no llevamos el objeto objno, se imprime el mensaje del Sistema 28 ("No llevo eso.") y una acción NEWTEXT y DONE son ejecutados.

Si el objeto en cuestión no es una prenda (Como se especifica en la sección /OBJ), se imprime el mensaje del Sistema 40 ("No puedo ponerme eso.") y una acción NEWTEXT y DONE son ejecutados.

En caso contrario, el objeto es movido a la localidad de objetos puestos (253), el flag 1 es decrementado en una unidad, y el mensaje del sistema 37 ("Ahora llevo puesto _.") es mostrado.

REMOVE objno.

Si el objeto objno lo llevamos o bien está en la localidad actual (pero no puesto), se imprime el mensaje del Sistema 50 ("No llevo puesto _.") y una acción NEWTEXT y DONE son ejecutados.

Si el objeto objno no está en la localidad actual, se imprime el mensaje del Sistema 23 ("No llevo puesto eso.") y una acción NEWTEXT y DONE son ejecutados.

Si el objeto objno no es una prenda, se imprime el mensaje del Sistema 41 ("No puedo quitarme _.") y una acción NEWTEXT y DONE son ejecutados.

Si el máximo de objetos que el jugador lleva (flag 1) es mayor o igual que el número de objeto llevables (flag 37), se imprime el mensaje del Sistema 42 ("No puedo quitarme _. Mis manos están llenas.") y una acción NEWTEXT y DONE son ejecutados.

En caso contrario la posición del objeto en cuestión es cambiada a llevado (localidad 254), el flag 1 es incrementado, y se muestra el mensaje del sistema 38 ("Me he quitado _.")

CREATE objno.

La posición del objeto objno pasa a ser la localidad actual. El flag

1 se decrementa si antes de eso el objeto lo llevaba el jugador.

DESTROY objno.

La posición del objeto objno pasa a ser la localidad de objetos no creados (252). El flag 1 se decrementa si antes de eso el objeto lo llevaba el jugador.

SWAP objno 1 objno 2

La posición de ambos objetos es intercambiada. El flag 1 no se ve afectado. El objeto referenciado actual pasa a ser el objeto objno2.

PLACE objno. locno+

La posición del objeto objno pasa a ser la localidad locno+. El flag 1 se decrementa si antes de eso el objeto lo llevaba el jugador, y es incrementado si el locno+ es 254 (llevado).

PUTO locno+

La posición del objeto al que se refiere actualmente (el objeto a cuyo número esté en la bandera 51) se cambia para ser el número de localidad de locno. La bandera 54 permanece cono su vieja localidad. La bandera 1 se decrementa si el objeto era llevado, y se incrementa si el objeto se pone en la localidad 254 (llevado).

PUTIN objno. locno.

1. Si el objeto con número de locno. no existe o no está marcado como un contenedor (opción C en el menú de Objetos Peso) entonces se genera un error de "Argumento Ilegal".
2. Si el objeto al que se refiere objno. es llevado puesto encima, entonces el Mensaje del Sistema 24 "No puedo, llevo puesto _" se imprime y las acciones NEWTEXT y DONE se ejecutan.
3. Si el objeto al que se refiere objno. está en la localidad actual (pero ni se lleva puesto encima, ni se lleva) el Mensaje del Sistema número 49 "No tengo _" se imprime, y las acciones NEWTEXT y DONE se ejecutan.
4. Si el objeto al que se refiere objno. no está en la localidad actual, pero tampoco está llevado, entonces un Mensaje del Sistema número 28 "No tengo uno de esos" se imprime, y las acciones NEWTEXT y DONE se ejecutan.
5. De cualquier otra forma, la posición del objeto al que se refiere objno. se cambia a la localidad a la que se refiere locno. La bandera 1 se decrementa y el Mensaje del Sistema 44 "El _ está en la", viene entonces una descripción de la localidad del objeto y luego el Mensaje del Sistema 51 ".", se imprime.

TAKEOUT objno. locno.

1. Si el objeto al que se refiere locno. no existe o no está marcado como un contenedor (la opción C en el menú de objetos y pesos) entonces se genera un error de "Argumento Ilegal".
2. Si el objeto al que se refiere objno. se lleva puesto encima o es llevado, el Mensaje del Sistema número 25 "Ya tengo _" se imprime y las acciones NEWTEXT y DONE se ejecutan.
3. Si el objeto al que se refiere objno. está en la localidad actual, el Mensaje del Sistema número 45 "_ no está en la", viene una descripción del objeto que va a ser usado como localidad, y luego el Mensaje del Sistema 51 "." se imprime y las acciones NEWTEXT y DONE se ejecutan.
4. Si el objeto al que se refiere objno. no está en la localidad actual y no está en la localidad locno., entonces el Mensaje del Sistema 52 "No hay uno de esos en", viene una descripción del objeto localidad y luego el Mensaje del Sistema 51 se imprime y las acciones NEWTEXT y DONE se ejecutan.
5. Si el objeto al que se refiere locno. no se lleva o no está puesto y el peso total de los objetos llevados por el jugador, más el objeto al que se refiere objno. excede del peso máximo llevable (bandera 52), entonces el Mensaje del Sistema 46 "El _ pesa mucho para mí" se imprime y las acciones NEWTEXT y DONE se ejecutan.
6. Si el máximo número de objetos ya se lleva (o sea, que la bandera 1 es mayor o igual que la bandera 37), el Mensaje del Sistema 27 "No puedo llevar más cosas" se imprime y las acciones NEWTEXT y DONE se ejecutan. En adición, cualquier bucle DOALL se cancela.
7. De otro modo, la posición del objeto al que se refiere objno. cambia a llevado, la bandera 1 se incrementa, y el Mensaje del Sistema 36 "Ahora tengo” se imprime.

   Nota Importante: No se hace ningún chequeo ni para PUTIN ni para TAKEOUT de qué objeto utilizado cano locno. esté actualmente presente. Esto debes hacerlo tú si es necesario.

DROPALL

Todos los objetos que sean llevados o puestos encima se crearán en la localidad actual (lo que dará el efecto de que todos los objetos han sido dejados caer) la bandera 1 se pondrá a 0. Este comando se ha incluido para compatibilidad con los antiguos sistemas, (léase QUILL). Es de notar que un DOALL 254 llevará a cabo un verdadero DROP ALL, y además tendrá en cuenta cualquier otra acción especial que se incluya.

Las siguientes seis acciones tratan de versiones automáticas como COGER, DEJAR, LLEVAR PUESTOS, QUITAR DE ENCIMA, PONER DENTRO y SACAR DE, (GET, DROP, WEAR, REMOVE, PUTIN and TAKEOUT). Son automáticos

por el hecho de que en vez de necesitar que se especifique el número del objeto, ellos convertirán el primer Nombre con su Adjetivo en el objeto actualmente indicado o referenciado, buscándolo en la tabla de Objetos Palabras. La búsqueda se hace por un objeto que esté en una de las siguientes localidades en orden descendente de prioridad. (Mirar las descripciones individuales). Esta búsqueda con prioridades permite que DAAD "sepa" de qué objeto se trata, si más de un objeto tiene el mismo nombre descriptivo (cuando el jugador no haya especificado un Adjetivo), en la localidad actual llevado o puesto encima o en un contenedor (si se trata del comando TAKEOUT).

AUTOG

Una búsqueda del número del objeto representado por Nombre (Adjetivo) 1 se hace en la tabla de Objetos Palabra según su prioridad de localización que será: aquí, llevado, puesto, etc. Es más lógico que el jugador esté tratando de coger un objeto que esté en la localidad actual, que uno que lleve o tenga puesto encima, por eso el orden de prioridades. Si el objeto es encontrado, su número se pasa a la acción GET. De otra forma, si es un objeto que existe en cualquier otra parte del juego o si el Nombre1 no está en el Vocabulario, el Mensaje del Sistema número 26 "No hay uno de esos aquí" se imprime. Si no, el Mensaje del Sistema número 8 "No puedo hacer eso" se imprime (significa que no es un objeto válido pero que existe en el juego). De todas formas, las acciones NEWTEXT y DONE se ejecutan.

AUTOD

Una búsqueda por el número del objeto representado por Nombre (Adjetivo) 1 se hace en la tabla de Objetos Palabras según la prioridad de localidad: llevado, puesto encima, aquí. Por ejemplo, lo más probable es que el jugador esté intentando dejar caer (DROP) un objeto que lleve, que lleve puesto encima o esté aquí. Si se encuentra un objeto, su número se pasa a la acción DROP. De otra forma si hay un objeto que existe en cualquier parte del juego, o si el Nombre1 no está en el Vocabulario, el Mensaje del Sistema 28 "No tengo uno de esos" se imprime. Si no, el Mensaje número 8 "No puedo hacer eso" se imprime, es decir (no es un objeto válido, pero sí que existe en el juego). De cualquier forma, las acciones NEWTEXT y DONE son ejecutadas.

AUTOW

En una búsqueda del número del objeto representado por Nombre (Adjetivo) 1 se hace en la tabla de objetos en orden de prioridad de: llevado, puesto encima, aquí. Es más probable que el jugador esté intentando ponerse encima (WEAR) un objeto que lleve en sus manos, que uno que lleve puesto o que uno que esté aquí. Si el objeto se encuentra, su número se pasa a la acción WEAR. De otra manera, si hay un objeto en existencia en cualquier parte del juego o si el Nombre1 no estaba en el Vocabulario, entonces el Mensaje del Sistema número 28 "No tengo uno de esos" se imprime. Si no, el Mensaje del

Sistema número 8 "No puedo hacer eso" se imprime. No es un objeto válido, pero sí existe en el juego. De cualquier modo, las acciones NEWTEXT y DONE se ejecutan.

AUTOR

Una búsqueda por el número del objeto representado por el Nombre (Adjetivo) 1 se hace en la tabla de Objetos Palabra can una prioridad de: puesto encima, llevado, aquí. El jugador, lo más probable es que esté tratando de quitarse (REMOVE) un objeto que lleve puesto, que uno que lleve en sus manos o que esté aquí. Si un objeto se encuentra, su número se pasa a la acción REMOVE. Si hay un objeto en existencia en cualquier parte del juego o si el Nombre1 no estaba en el Vocabulario, entonces el Mensaje del Sistema 23 "No llevo uno de esos" se imprime. Si no, el Mensaje del Sistema número 8 "No puedo hacer eso" se imprime (no es un objeto válido, pero existe en el juego). De cualquier forma, las acciones NEWTEXT y DONE se ejecutan.

AUTOP locno.

Una búsqueda del número del objeto representado por el Nombre (Adjetivo) 1 se hace en la tabla de Objetos Palabras en el siguiente orden de prioridad: llevado, puesto, aquí. Es más factible que el jugador esté intentando poner un objeto que lleva en sus manos dentro de otro, que uno que lleva puesto encima o está aquí. Si el objeto se encuentra, su número se pasa a la acción PUTIN. Si hay un objeto en existencia en cualquier parte del juego o si el Nombre1 no está en el Vocabulario, entonces el Mensaje del Sistema 28 "No tengo uno de esos" se imprime. Si no, el Mensaje del Sistema número 8 "No puedo hacer eso" se imprime. No es un objeto válido, pero existe en el juego. De todos modos, las acciones NEWTEXT y DONE son ejecutadas.

AUTOT locno.

Una búsqueda por el número del objeto representado por el Nombre (Adjetivo) 1 se hace en la tabla de Objetos Palabras según la siguiente prioridad: un contenedor, llevado, puesto, aquí. El jugador es más probable que esté intentando sacar un objeto que esté dentro de un contenedor que un objeto que sea llevado, puesto encima, o aquí. Si un objeto se encuentra, su número se pasa a la acción TAKEOUT. Si hay un objeto en existencia en cualquier parte del juego o si el Nombre1 no está en el Vocabulario entonces el Mensaje del Sistema "No hay uno de esos en él", viene luego una descripción del objeto que sirve de localidad, y luego el Mensaje 51 del sistema ".” se imprime. Si no, el Mensaje del Sistema 8 "No puedo hacer eso" se imprime. (No es un objeto válido, pero existe en el juego). De cualquier modo, las acciones NEWTEXT y DONE se ejecutan.

Nota Importante: No se hace ningún chequeo, ni por AUTOP ni por AUTOT, de que el objeto que se va a usar como contenedor, o sea el de locno. esté actualmente presente. Esto debe ser hecho por ti, si así es requerido.

COPYOO objno 1 objno 2

La posición del objeto al que se refiere objno2 se hace la misma que la posición en que estaba el objeto objno1. El objeto que se toma actualmente como referencia se pone en objno2.

RESET

Esta acción no se parece a la de igual nombre de PAW. Tiene la función de poner todos los objetos en la posición inicial. Además, establace todos los flags que tienen relación con el número de objetos llevados.

Acciones que cambian datos de objetos a flags y viceversa

COPYOF objno. flagno.

La posición del objeto al que se refiere objno. se copia en el flagno. de la bandera. Esto puede ser usado para examinar la localización de un objeto en comparación con otro valor de la bandera.
Por ejemplo: [COFYOF 1 11 SAME 11 38] puede ser usado para chequear que el objeto 1 esté en la misma localidad del jugador, aunque ISAT 1 255 sería mucho mejor.

COPYFO flagno. objno.

La posición del objeto al que se refiere objno. se pone como contenido del flagno. de la bandera. Una intención de copiar de una bandera que contenga 255 resultará en un error de "Argumento Inválido". El poner un objeto en una localidad inválida será aceptada puesto que no acarrea ningún peligro para las operaciones de DAAD. Esto debes tenerlo en cuenta (y no hacerlo) tú.

WHATO

Una búsqueda para el número del objeto representado por el Nombre (Adjetivo) 1 se hace en la tabla de Objetos Palabras en la siguiente prioridad: llevado, puesto, aquí. Esto es, porque se supone que cualquier uso del comando WHATO se referirá más que todo a objetos llevados que a cualquier objeto que se lleve puesto o esté aquí. Si un objeto se encuentra su número se pone en la bandera 51, lo mismo que con cualquier parámetro del objeto al cual se refiera actualmente en las banderas 54 a 57. Esto te permite crear otras acciones de tipo automático (en el manual de introducción dábamos el ejemplo de esto dejando caer objetos desde el árbol).

SETCO objno.

Establace objno. como el objeto referenciado actual.

WEIGH objno. flagno.

El verdadero peso al que se refiere objno. se calcula (por ejemplo: si es un contenedor, cualquier objeto que lleve dentro hará aumentar su peso, no te olvides de que puede haber varios contenedores uno dentro de otro y todos sus pesos se pueden añadir hasta el nivel 10) y el valor que dé, se pone en el flagno. de la bandera. El máximo es 255. Si el objeto al que se refiere objno. es un contenedor de peso cero, el flagno. de la bandera se pondrá a cero, puesto que cualquier objeto que esté en un contenedor que pese cero se considera que tiene un peso de cero (Sería un dispositivo antigravedad en una aventura de ciencia ficción).

Acciones para manipular los valores de flags

SET flagno.

El flag flagno pasa a valer 255.

CLEAR flagno.

El flag flagno pasa a valer 0.

LET flagno. value

El flag flagno pasa a valer el "value" especificado.

PLUS flagno. value

El flag flagno es incrementado en el "value" especificado. Si el resultado excede de 255, pasa a valer 255.

MINUS flagno. Value

El flag flagno es decrementado en el "value" especificado. Si el resultado es menor de 0, pasa a valer 0.

ADD flagno 1 flagno 2

El flag flagno2 es incrementado en el valor del flag flagno1. Si el resultado excede de 255, el flag flagno2 pasa a valer 255.

SUB flagno 1 flagno 2

El flag flagno2 es decrementado en el valor del flag flagno1. Si el resultado es menor que 0, el flag flagno2 pasa a valer 0.

COPYFF flagno 1 flagno 2

El valor del flag flagno2 pasa a ser el mismo que el del flag flagno1.

COPYBF flagno1 flagno2

El valor del flag flagno1 pasa a ser el mismo que el del flag flagno2. Es decir, es como COPYFF pero en sentido inverso.

RANDOM flagno.

El flag flagno pasa a tener un valor aleatorio entre 1 y 100. Puede ser útil para realizar acciones aleatorias en una manera más flexible que la que permite CHANCE.

MOVE flagno.

Esta es una acción muy poderosa diseñada para manipular PSI's (PNJs). Permite que la sentencia lógica actual se evalué contra la tabla de conexiones de la localidad indicada en el valor del flag flagno.

Si el verbo es encontrado (por ejemplo, el verbo era NORTE y hay una salida al norte), entonces el flag se actualiza para contener la localidad a la que esa conexión lleva.

Si no se encuentra el verbo, o el número de localidad original no era válido, entonces PAW salta a la siguiente entrada si la hay.

Esta funcionalidad puede usarse para facilitar que los personajes se muevan aleatoriamente, haciendo que el flag del verbo actual se modifique a un valor aleatorio, y luego tratar de hacer MOVE a ver si puede moverse. Nótese que cualquier movimiento que no esté en la tabla de conexiones no podrá realizarse con MOVE.

Acciones para manipular los flags que afectan al jugador

GOTO locno.

Cambia la localidad actual del jugador. Es equivalente a LET 38 locno.

WEIGHT flagno.

Calcula el peso real de los objetos llevado y puestos (esto es, calcula también lo que pesan los objetos dentro de contenedores llevados) hasta un máximo de 255. Dicho valor se almacena en el flag flagno.

Podría ser útil para determinar si un jugador puede cruzar un puente sin que se hunda, por ejemplo.

ABILITY value 1 value 2

Hace que el flag 37, máximo de objetos llevables, pase a valer "value1", y el flag 52, máximo objeto llevable, pase a valer "value2".

No se realizan comprobaciones para asegurarse de que el jugador no lleve ya más del nuevo máximo.

Equivale a
LET 37 value1
LET 53 value2

Acciones para manipular los flags del modo y formato de pantalla

MODE option

Permite cambiar el modo de operación de la ventana actualmente seleccionad. Lo valores indicados aquí deben ser sumados para cada opción que se desee, y facilitados en el parámetro option.

1 – Usa el set de caracteres superior (los 128 caracteres más altos del set de caracteres). Equivale a un #g permanente, como veréis después en el apartado de escape chars.

2 – No aparecerá el mensaje del sistema 32 ("Más...") si la ventana se llena.

Por ejemplo, MODE 3 hace que no salga el 'Más...' y además usa el set de caracteres superior.

INPUT stream option

El parámetro stream hará que el input (la entrada de órdenes) vaya a la ventana especificada. Un valor 0 no provocará que se use la ventana 0 sin embargo, sino la ventana actualmente seleccionada.

El valor de option se obtiene sumando las opciones siguientes que se quieran aplicar:

1 – Borrar ventana tras el input
2 – Escribir el input en la ventana actual cuando termine,
4 – Recuperar el texto que teníamos escrito tras un timeout.

TIME duration option

Permite que la entrada de órdenes del jugador tenga un "timeout" (demasiado tiempo esperando una orden) tras una duración especificada en segundos. En consecuencia, el sistema seguirá procesando las tablas sin esperar al jugador.

El parámetro 'option' indica si esto debe ocurrir también durante un ANYKEY o cuando sale un "Más..." porque se llena la ventana de texto. Para calcular el número que hay que poner en option, suma los valores de la siguiente tabla que quieres que ocurran:

1 – Solo ocurre cuando en el input no se ha tecleado nada aún.
2 – Ocurre también en "Más...".
4 – Ocurre cuando esperamos tecla en un ANYKEY.

Por ejemplo, TIME 5 6 (option = 2+4) permite 5 segundos sin teclear nada entre cada pulsación de tecla, y que ocurra en el ANYKEY, mientras que TIME 5 3 (option = 1+2) sólo permite que ocurra antes de teclear algo y en el "Más...".

TIME 0 0 deshabilita los timeouts (que es lo que ocurre por defecto)

Acciones para controlar la salida de pantalla

WINDOW window

Selecciona la ventana actual (0-7) como stream de salida de datos.

WINAT line col

Define la Ventana actualmente seleccionada, determinando donde comienza su esquina superior izquierda con los valores de línea y columna. Si la ventana tenía definido un ancho y un alto que hacen que exceda la pantalla, será recortada para tener un tamaño que quepa.

WINSIZE height width

Define el alto y ancho de la Ventana actualmente seleccionada. Si la Ventana excede el tamaño de la pantalla será recortada para caber.

CENTRE

Se asegura de que la ventana actual está centrada para el ancho de columna de la maquina actual. No afecta a la posición de la línea.

CLS

Borra la ventana actual.

SAVEAT
BACKAT

Guarda y restaura respectivamente la posición de escritura. Esto permite que mantener una posición mientras escribes en algún otro sitio en la misma ventana.

PAPER colour
INK colour

Selecciona el color de fondo o de tinta respectivamente. Colour depende de la máquina, y puede configurarse según el software que

viene con el DAAD original para crear los gráficos vectoriales. Dado que DAAD Ready no usa este modelo de gráficos, recomendamos que simplemente se utilice #ifdef (ver más adelante) para definir colores, dado que el uso de DG y otras herramientas de gráficos, es complejo.

BORDER colour

Cambia los colores del borde de la pantalla. Es específico por máquina.

PRINTAT line col

Cambia la posición de escritura de la Ventana actual para que sea la indicada por line y col. Si las coordenadas no caben dentro de la ventana, se cambia la esquina superior izquierda de la ventana.

TAB col

Mueve la posición de escritura actual a la columna indicada. Es como hacer un PRINTAT siendo line "la línea actual".

SPACE

Simplemente escribe un espacio. Más corto y ocupa menos que MES " ".

NEWLINE

Imprime un retorno de carro.

MES mesno. o MES "string"

Imprime el mensaje dado por mesno, o bien el mensaje entre comillas, en los colores actuales

Ten en cuenta que para la cadena de texto puedes usar comillas simples o dobles indistintamente, siempre que cierres y abras de igual modo.

MESSAGE mesno. o MESSAGE "string"

Similar a MES pero tras imprimir el mensaje ejecuta un NEWLINE.

SYSMESS sysno. o SSYMESS "string"

Similar a MES pero usa la tabla de mensajes del sistema para imprimir el numero sysno, o usa la cadena de texto directamente.

DESC locno. O DESC "string"

Similar a MES pero usa la tabla de descripciones de localidad para imprimir el numero locno, o usa la cadena de texto directamente.

Acciones para imprimir valores de flags en pantalla

PRINT flagno.

El contenido en decimal de la bandera a que se refiere flagno. se imprime con los colores actuales sin dejar espacios ni antes ni después. Esta es una acción muy eficaz. Por ejemplo, si la bandera 100 contiene el número de monedas que lleva el jugador, entonces una entrada en la tabla de Procesos del tipo MES "Tienes " y después PRINT 100 MESSAGE " monedas de oro.", se usaría para que se imprimiera el número de monedas que tiene.

```
MES "Tienes "
PRINT 100
MESSAGE " monedas de oro."
```

DPRINT flagno

Escribirá los contenidos de flagno y flagno+1 como un valor de dos bytes, generando un número en el rango 0-65535.

El contenido es generado multiplicando el valor del flag flagno+1 por 256 y sumando el valor del flag flagno.

DPRINT 255 carece de sentido, y producirá un valor sin sentido.

Acciones que manejan listados de objetos en pantalla.

Están controlados por el flag del sistema 53 (ver más adelante).

LISTOBJ

Si cualquier objeto está presente, entonces el mensaje del sistema número 1 "Además puedo ver" se imprime, seguido de una lista de los objetos presentes en la localidad actual. Si no hay ningún objeto, entonces no se imprime nada.

LISTAT locno+

Si hay algunos objetos presentes, se listan. Si no, el mensaje del sistema número 53 "Nada" se imprime. Hay que tener en cuenta que

usualmente hay que preceder esta acción con un mensaje del tipo de "En la bolsa hay", etc.

Acciones para grabar o cargar el estado actual del juego

0 – Acción normal
1 – Grabar a cinta (no preguntar "disco o cinta")
2 – Grabar a disco (ídem)

SAVE opt

Esta acción graba el estado actual del juego en disco o cinta. El mensaje del sistema 60 ("Escribe el nombre del fichero:") se imprime, y la rutina de input es llamada para que el jugador escriba el nombre. Si el nombre facilita no es válido se muestra el mensaje del sistema 59 "Error en nombre de fichero". En las máquinas de 8 bits este dato no es comprobado, simplemente hace que el fichero sea aceptabla.

LOAD opt

Esta acción carga una partida grabada de disco o de cinta. Se pide un nombre de fichero igual que con SAVE. Una variedad de errores puede ocurrir en cada máquina si el fichero no se encuentra (generalmente 'Error de E/S'). Si la carga tiene éxito la siguiente acción es ejecutada, en caso contrario una limpieza del sistema, GOTO 0, RESTART es ejecutado (dando como resultado en general el reinicio del juego).

RAMSAVE

De una manera similar a SAVE , la información de progreso en el juego es grabada, pero no en disco o cinta, sino en la memoria del ordenador. Este buffer es por supuesto volátil, y desaparecerá si apagamos el ordenador.

RAMLOAD flagno.

Esta acción restaura los datos grabados por RAMSAVE. El parámetro indica cual es el último flag restaurado, lo cual puede usarse para preservar valores a través de una recargar. Por ejemplo con una entrada así:

```
> RAMLO _
 COPYFF 30 255
 RAMLOAD 254
 COPYFF 255 30
 DESC
```

Que permite guardar la puntuación actual a pesar del RAMSAVE/RAMLOAD.

Nota 1: las acciones “RAM” pueden ser usadas para implementar OOPS, la acción que permite deshacer la última acción, creando una entrada en el bucle principal que haga RAMSAVE a cada ciclo.

Nota 2: Las cuatro acciones de grabar/cargar permiten el siguiente condActo ejecutarse, normalmente deben ir seguidas de un “RESTART” o al menos “DESC @fPlayer” para que el juego muestre los cambios.

Acciones que permiten pausar el juego

ANYKEY

El mensaje del sistema 16 ("Pulsa una tecla para continuar") se muestra, y el teclado es comprobado hasta que una tecla es pulsada, u ocurre un timeout (si habilitados en ANYKEY).

PAUSE value

Pausa por un valor/50 segundos (Pause 50 = 1 segundo). Sin embargo, si el valor es cero entonces la Pausa es por 256/50 segundos. Hay que destacar que el teclado se desconecta durante la duración de una pausa.

Acciones para el control del analizador sintáctico (parser)

PARSE n

El parámetro ‘n’ controla el nivel de indentación que debe buscarse. En este momento solo se soportan dos valores, siendo:

0 – Analiza el input principal para conseguir una SL.
1 – Analiza cualquier cadena metida entre comillas dobles (“) para que estuviera contenida en la SL extraída.

El modo 0 es el método primario para convertir las órdenes del jugador en sentencias lógicas (SL).

El modo 1 se ha diseñado para manejar PSIs (PNJs). Cualquier cadena metida en comillas que es encontrada en el input es convertida en la SL, reemplazando la frase principal.

Si no hay frase presente a nivel 0 entonces se llama a la rutina de input para pedir nueva frase, siempre que no hubiera un buffer de ordenes previo del que sacar más SLs.

En nivel 1 y superior el siguiente condActo es ejecutado. Esto ocurre en todos los niveles si la SL es inválida.

Nótese que DAAD mirará el siguiente condActo en lo que puede ser considerado una situación de “fallo”, lo cual es diferente a lo

habitual. En otro caso, se pasa a la siguiente entrada.

Dado que la SL actual es modificada, debe ser gestionado con cuidado.

Por ejemplo, el mínimo proceso 0 sin inicialización podría ser:

```
> _ _
 PARSE 0
 MESSAGE "No te entiendo"
 REDO

> _ _
 PROCESS x ; Maneja cualquier orden
 REDO
```

Para usarlo para hablar con PSIs PNJs) habrá dos llamadas más, en el proceso x del ejemplo anterior, que serían similares a:

```
> DECIR nombre
  SAME pos 38 ;Está aquí?
  PROCESS y ;Decodificar lo dicho…
  DONE ;LS destruida, así que siempre DONE

> DECIR nombre
  MESSAGE z ;"No está aquí!"
  DONE
```

Con un proceso x similar a:

```
 > _ _
  PARSE ;Siempre ejecuta esto
  MESSAGE x ;"No te entiende"
  DONE

> word word
  CondAct list ;Cualquier frase que entienda

> _ _
  MESSAGE x ;como arriba o un mensaje diferente, porque es una frase que
contiene cosas del vocabulario, pero ninguna esperada
```

NEWTEXT

Fuerza que se pierda cualquier frase que permanezca en la línea actual de INPUT. Se debe usar para prevenir que el jugador continúe con un INPUT dañado, sin tener un nuevo INPUT, si algo sucede durante una situación. Por ejemplo, la acción COGER (GET) ya de por sí lleva un NEWTEXT si por alguna razón falla en coger el objeto que se requiere. Ello se usa para prevenir el desastre en una frase como: COGE LA ESPADA Y MATA A MANOLO CON ELLA por si se lleva a cabo un ataque a Manolo sin tener la espada, lo cual es muy peligroso.

SYNONYM verb noun

Sustituye el verbo y/o nombre de la SL actual por los dados. Si uno de los dos es un guion bajo "_" ese no se sustituye

Ejemplo:

```
> ENCENDER LUZ
  SYNONYM PULSAR INTERRUPTOR

> PULSAR INTERRUPTOR
  ; Acciones y condiciones...
```

Nota importante: SYNONYM es una acción, por lo que el estado de "hecho" del proceso en curso queda activado por su ejecución. Será necesario que haya por tanto un match posterior en el proceso, o podría resultar el proceso "hecho" sin aparentemente haber hecho nada.

Entre los "CondActos Maluva" que veremos más adelante, hay uno llamado XUNDONE que permite deshacer el estado de "hecho", que en ocasiones puede ser interesante usar después de SYNONYM.

Acciones de saltos, bucles y subrutinas

PROCESS procno.

Esta poderosa acción transfiere la atención de DAAD hacia el número de tabla de Proceso especificado por procno. Este subproceso exhibirá las mismas características que la tabla que lo llamó. Por ejemplo, si ha sido llamado desde la tabla de Respuestas, DAAD seguirá buscando la correlación del Verbo y el primer Nombre de la SL y las palabras que hayan sido entradas en la tabla principal. Hay que darse cuenta de que es una verdadera llamada a una subrutina y de que cualquier salida desde la tabla nueva (por ejemplo, causada por un DONE, OK, etc.) llevará de nuevo el control al siguiente condActo

que esté en lista después de la acción PROCESS. Un subproceso puede ser llamado (anidar) a otros procesos hasta una profundidad de 10 en cuyo punto se creará un error de "Límite Alcanzado".

REDO

Reinicia el procesado de la tabla actual.

DOALL locno+

Otra acción muy poderosa que permite la mejora de cualquier comando de tipo "ALL" (todo).

1. Se intenta encontrar un objeto en la localidad a la que se refiere locno. Si no se encuentra (no es satisfactorio) el DOALL se cancela, el mensaje del sistema 8 "No puedes hacer eso es mostrado", y una acción DONE se ejecuta.

2. Si se encuentra un objeto, el número del objeto se convierte en el Nombre1 de la SL (y el Adjetivo1 si está presente) por medio de una referencia a la tabla de Objetos Palabras. Si Nombre (Adjetivo) 1 es similar o hace juego con Nombre (Adjetivo) 2 entonces se hace una vuelta al paso 1. Esto es para facilitar el "Verbo TODO EXCEPTO objeto" (coge todo excepto...).

3. El siguiente condActo y/o entrada en la tabla se considera entonces. Esto convierte efectivamente una frase del tipo "Verbo todo" en una del tipo "Verbo objeto" la cual será entonces procesada por la tabla como si el jugador lo hubiese tecleado de la última forma.

4. Cuando se hace un intento de salida de la tabla actual, si el DOALL todavía está activo (por ejemplo, no ha sido cancelado por una acción), entonces la atención del PAW vuelve de nuevo al bucle DOALL desde el paso 1. La búsqueda continuará desde el siguiente objeto (el que tenga un número más alto).

La consecuencia de este método de búsqueda a través de la tabla de Objetos Palabras es que los objetos que tengan el mismo Nombre y Adjetivo en su descripción, (y en donde el juego sólo sabe a cuál objeto se refiere por su presencia en esa localidad) son buscados en un orden ascendente de número de objeto, si no algunos de ellos se perderían.

Si se usa un DOALL para cosas como abrir todo, hay que tener en cuenta el hecho de que, en el funcionamiento de la mayoría de las puertas, se han puesto banderas y que entonces habría que convertir esas banderas en objetos para poderlas incluir en un bucle DOALL.

El intérprete DOS soporta un DOALL extendido donde los DOALL se pueden anidar si están en distintos procesos (por ejemplo, durante un DOALL, se hace una llamada a PROCESS, y ese proceso tiene otro DOALL). Para habilitar eso hay que llamar al intéprete on el parámero

-NDOALL (nested doall). No es my probable usar esto dado que otros intérpretes no lo soportan, pero es bueno saberlo.

SKIP distancia | label

Donde distancia es de -128 a 128, o una etiqueta.

Cambiará la ejecución al principio de una entrada en la misma tabla, donde -1 es la misma entrada, -1 la anterior, etc. También se puede saltar hacia delante, siendo 0 la siguiente entrada, 1 la subsiguiente, etc.

SKIP puede también aceptar como parámetro una etiqueta, que es un símbolo precedido por un signo de dólar ($), que debe colocarse justo antes de una entrada.

Utilizar etiquetas es más cómodo (no tienes que contar entradas) y además más fácil de mantener, que usar valores absolutos. Es más fácil de mantener porque contando entradas un día te puede cuadrar poner -4, pero luego vas y metes otra entrada por medio por cualquier razón, y ahora debería ser -5, pero como no estás viendo el SKIP no lo cambias y de repente te fallan cosas que antes funcionaban.

Ejemplo:

```
$saltoatras
> _ _
  PRINT Flag
  MINUS Flag 1
  NOTZERO Flag
  SKIP $saltoatras

> _ _
  ZERO Error
  SKIP $Adelante

> _ _
  EXIT 0

$ Adelante
> _ _
…
```

Acciones que salen de la ejecución de procesos

RESTART

Cancela cualquier bucle DOALL y cualquier llamada a sub-procesos, y salta para empezar a ejecutar el proceso 0 de nuevo.

END

Se muestra el mensaje del sistema 13 ("¿Quieres jugar de nuevo?") y se llama a la rutina de input. Cualquier DOALL y sub-procesos son cancelados. Si la respuesta no empieza con el primer carácter del mensaje del sistema 31 ("N") se salta a la inicialización. En caso contrario el jugador es devuelto al sistema operativo, haciendo un EXIT 0.

EXIT value

Si el valor el s0, se devuelve al jugador al sistema operativo (se sale del juego). Cualquier valor distinto a 0 reinicia el juego.

Nótese que a diferencia de RESTART que solo reinicia procesos, este procedimiento borra, reinicia ventanas, etc.

Acciones de salida de tabla

DONE

Esta acción salta al final del proceso en ejecución y marca el estado interno de "hecho" que le dice a DAAD que algo se ha hecho ya. No se evalúan más entradas de ese proceso ni más condActos de la entrada donde esté el DONE.

En consecuencia, se devolverá el control al proceso que llamó a ese proceso, o bien volveremos al punto de bucle si estamos dentro de un DOALL.

NOTDONE

Esta acción ejecuta un salto al final de la tabla de Procesos y le índica a DAAD que ninguna acción ha sido hecha. Quiere decir que no se consideran más condActos ni entradas. Causará una vuelta a cualquier tabla de Procesos previa o a un punto de partida de cualquier bucle DOALL que esté activo. Esto hará que DAAD imprima uno de los mensajes "No puedes" si es necesario, es decir, si no hay más acciones y no hay entradas presentes en las conexiones para el Verbo actual.

OK

El Mensaje del Sistema número 15 "Vale" o "Bien" se imprime y la

acción DONE se ejecuta.

Acciones para el manejo del sonido

BEEP value value

Realiza un pitido por el altavoz de duración tantos 1/50 de segundo como el primer valor diga, y de la frecuencia que dice el segundo.

Acciones para llamar a rutinas externas de DAAD

Estas acciones serían **EXTERN**, **CALL**, **SFX** y **GFX**. Sin embargo, dado que DAAD Ready gestiona internamente EXTERN, y las demás no son consideradas precisamente sencillas de utilizar, dejamos fuera del manual de DAAD Ready estas cuatro acciones. Si quieres más información la tienes en el manual original de DAAD.

Acciones para el manejo de gráficos vectoriales

Al igual que en el caso anterior, dado que DAAD Ready utiliza su propio sistema de gráficos, no vamos a detallar en este manual el uso de **PICTURE** y **DISPLAY**. Quizá tengas que utilizarlos si acabas añadiendo gráficos para DOS, ST, Amiga o PCW, pero te sugerimos mirar el manual original de DAAD.

CondActos Maluva

Maluva es una extensión de DAAD que está incluida de facto en DAAD Ready. Maluva gestiona muchas cosas y estandariza otras. Por ejemplo, usando Maluva puedes disponer de nuevos condActos que te permiten cargar imágenes desde disco, textos adicionales, etc.

Ten en cuenta que DRC (el nuevo compilador) conoce estos condActos, y en ocasiones los reemplaza automáticamente por otros en determinados destinos, por otros que soporta ese destino. Por ejemplo, si usas XPICTURE y el destino es un 16-bit o PCW, lo va a sustituir de manera automática por PICTURE, que hace casi la misma función.

Esta es una lista de los condActos Maluva soportados por cada máquina. Las máquinas de 16 bit no soportan ninguno.

-	ZX ESXDOS	ZX +3	CPC	C64	Plus/4	MSX	ZX Next	ZX Uno	PCW
XPICTURE	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Nativo
XSAVE	Sí	Sí	Nativo	Nativo	Nativo	Sí	Sí	Sí	Nativo
XLOAD	Sí	Sí	Nativo	Nativo	Nativo	Sí	Sí	Sí	Nativo
XMESSAGE	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
XPART	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
XBEEP	Nativo	Nativo	Sí	Nativo	Nativo	Sí	Nativo	Nativo	No
XSPLITSCR	No	No	Sí	Sí	No	No	No	Sí	No
XUNDONE	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí	Sí
XNEXTCLS	No	No	No	No	No	No	Sí	No	No
XNEXTRST	No	No	No	No	No	No	Sí	No	No
XSPEED	No	No	No	No	No	No	Sí	Sí	No

XPICTURE locno

Dibuja la imagen para la localidad locno en la parte superior de la pantalla.

XSAVE opt

Guarda la partida en curso en el disco. Por compatibilidad con SAVE mantiene el parámetro opt, pero es ignorado.

XLOAD opt

Carga la partida previamente grabada. El parámetro opt también es ignorado.

XMES "string"

Escribe la cadena entre comillas. Nótese que este condActo permite textos adicionales a los de los de las tablas de mensajes y los condActos MES, MESSAGE, SYSMESS y DESC, con lo que tu juego tiene una capacidad de escribir muchos más textos. XMES tiene las siguientes limitaciones:

- Cada mensaje no puede tener más de 511 caracters.
- Todos los mensajes usados por XMES y XMESSAGE no pueden superar, una vez comprimidos, los 64K.
- Estos xmessages son leídos desde discos a demanda, por lo que dependiendo del dispositivo de lectura, su tiempo de respuesta puede ser lento. Úsalos con cuidado. Esto también significa que en máquinas que usan un Split-mode, como C64, CPC y ZX-Uno, se pueden producir efectos extraños en pantalla mientras se leen del disco, aunque para ZX-Uno es poco probable, porque lee de una tarjeta SD muy rápida.
-

XMESSAGE "string"

Igual que XMES pero ejecuta un NEWLINE al final.

XPART value

Si tienes un juego de dos partes y quieres tener ambas en la misma cara de un disco, y ambas usan xmessages, hay un problema porque ambas tienen los xmessages guardados en ficheros con extensión XMB de igual nombre, y se pisarían.

XPART puede ejecutarse antes de cualquier XMES/XMESSAGE para que DAAD sepa que estamos en la parte 2, 3, etc.

- Para Spectrum y MSX, el parámetro "value" es añadido al 0 de 0.xmb, así que si el valor es 1, en lugar de cargar los mensajes desde 0.XMB los cargará desde 1.XMB

- Para C64 y CPC, si el valor no es 0, sumará 50. Así en lugar de cargar los mensajes desde 00.XMB, 01.XMB, etc, lo hará desde 50.XMB, 51.XMB, etc.

XSPLITSCR mode

Para las máquinas que soportan Split screen (actualmente CPC, C64 y ZX-Uno) este condActo define el modo en el que lo hace.

Esto es lo que "mode" significa para cada máquina:

Máquina		Modo parte superior	Modo Parte inferior
CPC	0	CPC Modo 1	CPC Modo 1
CPC	1	CPC Modo 0	CPC Modo 1
CPC	2	CPC Modo 2	CPC Modo 1
C64	0	C64 HiRes	C64 HiRes
C64	1	C64 Multicolor	C64 Hires
ZXUno	0	Standard	Standard
ZXUno	1	Timex HiColor con ULAPlus	Standard

XUNDONE

XUNDONE cambia el estado de "hecho" interno. Cada vez que DAAD ejecuta una acción, el intérprete actica el estado "hecho" interno. Eso significa que, si ISDONE o ISNDONE son ejecutadas, tendrán éxito o no dependiendo de ese estado.

Nótese que a diferencia de NOTDONE, XUNDONE solo borra ese estado, pero no va al final del proceso actual.

Hay casos en los que incluso cunado una acción se ejecuta en un proceso, no queremos que se marque el estado de hecho. Esto es muy claro en el caso de usar SYNONYM, porque por regla general, aunque SYNONYM se trata de una acción, en la práctica, y a ojos del jugador, realmente no hace nada.

```
> ENCENDER LUZ
    SYNONYM PULSAR INTERRUPTOR
```

Si más tarde pones esta entrada, y resulta que ya estaba pulsado, la entrada fallará:

```
> PULSAR INTERRUPTOR
 ZERO fInterruptorPulsado
 MESSAGE "Enciendes la luz."
 SET fInterruptorPulsado
 DONE
```

Sin embargo, como DAAD considera que ya se ha hecho algo, por ese SYNONYM, en lugar de llegar a un "No entiendo" nos pasaría esto:

```
> PULSAR INTERRUPTOR
> (pide nueva orden)
```

Para estos casos y otros, en los que pese haber algo "hecho" no nos interesa que se marque el valor interno de "hecho", está XUNDONE.

Nótese que no hay un condActo XDONE para marcar como "hecho", porque puedes hacerlo con cualquier condActo que no haga nada, como por ejemplo "GOTO @fPlayer".

XBEEP duration tone

- La duración es 1/50 de segundos, así que duración 50 es un segundo.
- El tone es un valor de 48 a 238, que permite tocar notas en 8 octavas diferentes. Los valores por encima o por debajo de esos son tomados como silencios.
-
- Solo son válidos los valores pares, así que Do es 48, Do# es 50, Re es 52, Re# es 54, Mi es 56, Fa es 58, etc. Si usas valores impares, probablemente solo oirás ruido.

Para cada octava estos son los rangos:

Octava	Rango
1	48-70
2	72-96
3	98-118
4	120-142
5	144-166
6	168-190
7	192-214
8	216-238

Para las máquinas que no soporta XBEEP, DRC cambiará estas llamadas por BEEP (que funciona igual).

XPLAY "string"

Aunque no es exactamente un condActo Maluva, sino un falso condActos, XPLAY permite tocar melodías usando BEEP en una manera más cómoda. La cadena esperada es exactamente como la de un Spectrum 128K, por ejemplo "CDE" para Do-Re-Mi. XPLAY básicamente soporta las mismas funcionalidades que un Spectrum 128K, pero:

1) Es para todas las máquinas de 8 bit excepto PCW. No es una funcionalidad solo para Spectrum.
2) XPLAY en realidad será convertida por el compilador a un montón de XBEEP seguidos, y estos beeps los tocará cada máquina como lo suela hacer, por ejemplo, será por el speaker en el Spectrum, y por el chip AY en CPC.
3) Solo soporta un canal

Como el formato de las cadenas es complicado de explicar, os recomendamos buscar el detalle de la orden PLAY en cualquier manual de Spectrum 128k.

XNEXTCLS

Esta funcionalidad es solo para el Spectrum Next. Como las imágenes en el Next son pintadas en su "layer2", un CLS normal no funciona, así que si quieres borrar las imágenes, debes usar esta función.

XNEXTRST

También exclusive para el Next. Es usada para realizar parte de la función que hace el condActo END, que por la misma razón que el CLS, no funciona como debería. Mira la sección "finalizando juegos de Next" al final de este manual.

XSPEED value

Esta función hace que la CPU del ZX-Uno y el ZX-Spectrum Next pase a estar a 7Mhz si el valor 1, volviendo al estándar 3.5Mhz con el valor 0. Otros valores son ignorados.

Incrementar la velocidad hace que los textos y los gráficos salgan más rápido, pero también afecta a XPLAY, XBEEP y BEEP haciendo que suenen más agudos, por lo que recomendamos pasar a velocidad normal antes de estos dos condActos.

Control de errores en condActos Maluva

Algunas funciones de Maluva pueden fallar por razones externas. Por ejemplo, leer una imagen desde disco puede fallar porque el fichero no esté, porque el usuario sacó el disco, etc. DAAD y Maluva no hacen nada aparentemente en esos casos, básicamente no pintan la imagen, o no pintan el texto si era un XMESSAGE y no están los ficheros.

Pero sí que pasa algo internamente: cuando estas funciones fallan, el bit 7 del flag 20 se pone a 1, y cuando tienen éxito el mismo bit se pone a 0. Así, tras un XPICTURE se puede hacer un "GT 20 127", y la condición tendrá éxito si la carga falló, lo que permite dar mensajes como "Por favor introduzca el disco".

- ```
  _ _
  XPICTURE @38
  GT 20 127
  MESSAGE "Aviso: imagen no encontrada, por favor meta el disco"
  ```
  ```

O incluso esperar a que el disco se meta:

```
_ _
XPICTURE @38
GT 20 127
MESSAGE "Por favor pulse una tecla cuando haya metido el disco.”
ANYKEY
SKIP -1
```

Por otro lado, Maluva puede también reportar este tipo de errores mediante el estado de “hecho” interno

Si pones a 1 el bit 0 del flag 20 (LET 20 1), Maluva, aparte de grabar el resultado en el bit 7 del flag 20, marcará la acción como hecha o no dependiendo del resultado. Así puede usarse ISDONE e ISNDONE tras la llamada a XPICTURE o XMESSAGE:

```
_ _ AT 200
    LET 20 1
    XPICTURE 120
    ISNDONE
    MESSAGE "Error: picture not found, please insert disk"
```

DAAD para autores PAW o Quill

Los procesos

```
0 -> No lo toques
1 -> No lo toques
2 -> No lo toques
3 -> Hace la función del proceso 1 en PAW
4 -> Hace la función del proceso 2 en PAW
5 -> Hace la función de la tabla de respuestas en PAW
6 -> Proceso de inicialización, puedes poner tus propios mensajes o
imágenes aquí, o cualquier otro tipo de inicialización.
```

Objetos

DAAD permite atributos, lo cual significa que cada objeto tiene una serie de 16 valores que pueden estar activos (1) o inactivos (0). Los puedes ver en la sección /OBJ, y se puede comprobar su valor con los condActos HASAT y HASNAT. Con esto, por ejemplo, puedes definir un determinado puzle que permita cortar algo con objetos afilados, y determinar que el atributo 7 es "afilado" y ponérselo a 1 a una espada y a un cuchillo. Después, para resolver el puzle, puedes comprobar si el objeto que usa es afilado (con HASAT) en lugar de comprobar si lleva la espada o el cuchillo. También podrían hacerse objetos "voluminosos", que de llevarlos no puedas pasar por una grieta estrecha en una cueva, etc.

Indirección

Se ha explicado arriba, y no la soportaba PAW ni Quill.

Escribiendo mensajes

Mas que de DAAD, esta es una ventaja de DRC, el nuevo compilador. Como habrás visto en la descripción de MESSAGE, MES, SYSMESS y DESC, ahora puedes poner el mensaje en texto como parámetro, y así no preocuparte de los números de mensaje.

Además, Maluva facilita el condActo XMESSAGE (y XMES), que espera una cadena como parámetro, y que escribirá ese texto, pero lo almacenará en un fichero aparte. Esto permite 64K más de mensajes en un juego hecho con DAAD Ready Los XMESSAGES además no están limitados en 256, como con las otras tablas. Solo se acaban cuando se acaban los 64K extra.

Errores

Aunque hacemos lo posible por comprobar todo desde el compilador, algunos errores no se podrán detectar hasta que se esté probando el juego.

Los intérpretes pueden lanzar varios tipos de error. Generalmente será una pequeña Ventana en la esquina en los 8 bit y centrada en los 16 bit.

Los errores son:

- **I/O Error** → Error al cargar o grabar un fichero

- **BREAK** → El jugador pulsó break

- **Error n** → Esto es un error específico de esa máquina, es necesario mirar el manual de esa máquina para saber qué es el error "n".
- **Game Error n** →

0 – Número de objeto no válido
1 – Asignación ilegal a HERE (Flag 38)
2 – Intento de que la localidad pase a ser 255
3 – Demasiadas llamadas anidadas a procesos
4 – Se intenta anidar DOALL
5 – CondActo ilegal
6 – Llamada a proceso no válida
7 – Número de mensaje no válido
8 - PICTURE no valida (solo lo gráficos vectoriales, no aplica en *DAAD Ready*)

El analizador sintáctico (parser)

El parser trabaja escaneando cada entrada de texto del jugador para encontrar palabras que estén en el vocabulario, y extrae frases que puede convertir en sentencias lógicas (SL).

Cuando se extrae una frase, la tabla de respuestas y conexiones son analizada para ver si son reconocidas. Si no se muestra el mensaje del sistema 8 ("No puedes hacer eso.") o el 7 ("No puedes ir en esa dirección") dependiendo del valor del verbo (si el verbo es menor que el verbo 14, se usa el 7, porque es un verbo de dirección).

Una nueva orden se pide también si alguna acción falla de alguna manera, por ejemplo "pesa demasiado" al coger algo, para evitar situaciones catastróficas como que el jugador escriba "COGE ESPADA Y MATA TROLL" y ocurra esto:

- COGE ESPADA Y MATA TROLL

Llevas demasiado peso, no puedes coger la espada.
Intentas matar el troll con tus manos vacías, pero mueres en el intento.

Si una SL tiene éxito, entonces otra trata de extraerse si hay más texto en el buffer, o si no lo hay se le pide más órdenes al jugador.

Las frases se separan por las conjunciones del vocabulario (Y, ENTONCES) y por cualquier signo de puntuación ("DEJA LLAVE, COGE ESPADA Y MATA TROLL" son tres frases).

Las terminaciones pronominales -lo, -la, -los, etc. puede usarse para referirse al objeto de la frase anterior, incluso si no se escriben ambas frases a la vez. Por ejemplo: "COGE ESPADA Y LIMPIALA" o "COGE ESPADA" y después "LIMPIALA".

Los nombres con valores inferiores a 50 son nombres propios y no les afectarán las terminaciones pronominales.

Una sentencia lógica se configura así

```
(Adverbio)Verbo(Adjetivo1(Nombre1))(preposicion)(Adjetivo2(Nombre2))
```

Donde las cosas entre paréntesis son opcionales, por lo que la mínima frase es un verbo, o un nombre convertible a verbo (los nombre con valor menor de 20 que si no hay verbo en la frase se convertirán en verbo, por ejemplo, NORTE).

Si el verbo se omite, el analizador asumirá que la SL usa el verbo de la orden anterior, para que "COGER ESPADA Y ESCUDO" se convierta

de hecho en “COGER ESPADA” + “COGER ESCUDO”.

Nótese que la frase puede escribirse de mala manera y el parser aun entenderla, por ejemplo “LLAVE COGER” o “COGER LENTAMENTE LLAVE”.

Una orden real podría ser:

COGE TODO. ABRE LA PUERTA Y VE AL SUR ENTONCES COGE EL CUBO Y MIRALO”

de la cual saldrán 5 SLs:

COGE TODO
ABRE PUERTA (porque LA no está en el vocabulario)
SUR (porque VE no está tampoco)
COGE CUBO
MIRA CUBO (por la terminación -LO)

Nótese que DOALL no pasará por el objeto que coincida con el segundo nombre y adjetivo, lo que facilita una manera simple de hacer un “COGER TODO MENOS XXX”.

Español

Si el verbo es de menos de 6 letras tendrás que incluir sinónimos en el vocabulario que incluyan hasta 5. Si ya tiene 4 tienes que añadir una solo (TOMA -> TOMAL), si tiene 3 hay que añadir dos 'lo' and 'la' (PON -> PONLO, PONLA), y si tiene solo dos, hay que añadir todas las terminaciones (DA -> DALE, DALOS, DALA, DALAS, DALES)

El parser español maneja los nombres, pronombres y adjetivos de manera diferente al inglés. Específicamente, asume que el adjetivo va detrás del nombre.

Otros idiomas

A pesar de que DAAD solo soporta inglés y español, DAAD Ready tiene un soporte limitado de portugués y alemán. Eso significa que los caracteres de otros idiomas como “õ” o “ß” pueden mostrarse, pero no cambia el parser, así que por debajo de un juego en alemán está el parser de inglés, y de un juego portugués el de español. Tenedlo en cuenta para juegos en esos idiomas.

Ahora mismo no hay soporte para otros idiomas, pero debería ser posible con cierta facilidad dar soporte a cualquier idioma que use el juego de caracteres LATIN-1 (la mayoría de los idiomas europeos lo usan).

Los flags del sistema

Los flags normales están libres para ser usado por tus juegos, pero

si le echas un ojo a TEST.DSF, podrás ver que define un uso para cada flag en el rango 0-63, los flags del sistema. Además, define nombres simbólicos para dichos flags. Aunque DAAD no referencia todos, solo los que ves más abajo, podrían ser usados en futuras ampliaciones, así que mejor no usarlos.

La mejor manera de comprobar algunos flags que se usan a nivel de bit es usar HASAT. Por ejemplo, HASAT MOUSE se cumplirá si el sistema soporta ratón.

0	Cuando no es cero indica que el juego está oscuro (ver también el objeto 0)
1	Guarda la cantidad que el jugador lleva, pero no lleva puestos como prenda.
2 a 28	Estos flags no son usados por los intérpretes DAAD, por lo que en principio estarían libres para tus juegos. Por otro lado, el fichero TEST.DSF está usando el flag 28 para funcionalidades internas, mientras que Maluva usa el flag 20. Por tanto, no puedes usar esos flags, pero además recomendamos no usar ninguno de este bloque porque podrían ser usados para futuras extensiones. Nuestra recomendación es que empieces por usar el flag 254 para tus cosas, luego el 253, 252, etc. Así quedarás lejos de peligro.
29	Bit 0 - Mouse presente (16 bit solo).
30	Flag de puntuación (Score) – no es realmente usado directamente por DAAD pero es tradicional (de PAW).
31/32	(LSB/MSB) guarda el número de órdenes que el jugador ha dado.
33	Guarda el número del verbo de la sentencia lógica actual
34	Guarda el número del nombre de la sentencia lógica actual
35	Guarda el número del adjetivo de la sentencia lógica actual
36	Guarda el número del adverbio de la sentencia lógica actual
37	Guarda el número máximo de objetos que el jugador puede llevar (inicialmente 4), Se cambia usando ABILITY
38	Guarda la localidad actual del jugador
39/40	No usados
41	Da el número de stream para recibir el input. 0 significa el stream actual. Se usa en módulo 8, por lo que un valor de 8 es considerado un 0
42	Guarda el prompt para la entrada de órdenes del jugador. Si vale 0 selecciona uno aleatorio entre 4.
43	Guarda el número de la preposición de la sentencia lógica actual
44	Guarda el número del segundo nombre de la sentencia lógica actual
45	Guarda el número del segundo adjetivo de la sentencia lógica actual

46	Guarda el pronombre actual, el nombre al que sustituiría "IT" o la terminación "-lo","-la",
47	Guarda el adjetivo del pronombre
48	Guarda la duración de un timeout
49	Guarda los flags de control de timeout (nivel de bit): 7 – A 1 si ocurrió un timeout 6 – A 1 si había texto escrito que recuperar (no es de uso para el autor) 5 – Poner a 1 para forzar que el texto escrito se recuperar tras el timeout 4 – Pon a 1 para poner el input en el stream active tras editar 3 – Pon a 1 para borrar el input 2 – Pon a 1 para que un timeout pueda ocurrir durante un ANYKEY 1 – Pon a 1 para que pueda ocurrir en "Más..." 0 – Pon a 1 si un timeout solo puede ocurrir al principio del input (y no si el jugador ya ha tecleado algo)
50	Guarda el valor del objeto para el bucle DOALL, es decir, el valor que sigue al DOALL
51	Guarda el último objeto referenciado por GET/DROP/WEAR/WHATO etc. Es el número del objeto actualmente referenciado tal y como se muestra en muchos mensajes con el guion bajo.
52	Guarda la fuerza del jugador (máximo peso que puede llevar, inicialmente 10). Se establace con ABILITY
53	Guarda los flags de listar objetos Bit 7 – Se pone a 1 si al menos un fue listado en un LISTAT 6 – Ponerlo a 1 generará listados de objetos continuos, en lugar de uno encima de otro. LET 53 64 hará que DAAD liste los objetos en una línea generando una frase válida.
54	Guarda la localidad del último objeto referenciado
55	Guarda el peso del ultimo objeto referenciado
56	Guarda 128 si el último objeto referenciado es un contenedor
57	Guarda 128 si el último objeto referenciado es una prenda
58/59	Guarda los atributos de usuario del último objeto referenciado
60/61	Son los flags de teclado. Ver INKEY.
62	Es un flag usado para controlar los distintos modos de video que usan el intérprete de PC y de ST. Como DAAD Ready solo usa un modo en PC (320x200x16) y no genera para ST, no tiene uso.
63	Defines la ventana active en este momento. Nótese que es solo una copia del número, por lo que cambiarlo no tendrá efecto. Para cambiar de ventana activa usar

	WINDOW
64-254	Disponibles para tu propio uso
255	Se usaba en una versión anterior de TEST.DSF, así que mejor no usarlo.

El fichero Fuente (DSF)

El fichero fuente de DAAD consiste en varias secciones relacionadas que describen la Aventura. Las secciones son:

Secciones

La sección de Control (CTL)

Esta sección está obsoleta y no se usa ya. Debe haber un caracter "_" en ella, pero es por razones históricas.

La sección de Vocabulario (VOC)

Cada entrada en esta sección contiene una palabra (o las 5 primeras letras de una palabra), un valor numérico de la palabra, y un tipo. Las palabras que tienen el mismo número y tipo son consideradas sinónimas, y pueden usarse indistintamente.

La sección de Mensajes del Sistema (STX)

Esta sección contiene mensajes usados por los intérpretes, numerados del 0 al 255 (aunque no se usan todos). Ya hemos comentado en los condActos cuando se usan dichos mensajes.

Dado que no se usan todos los mensajes del Sistema, el resto pueden usarse para nuestra Aventura.

La sección de Mensajes de usuario (MTX)

Esta sección contiene los textos que será necesarios para la Aventura. Los mensajes se numeran del 0 al 255.

Con DAAD Ready no necesitas realmente añadir mensajes a esta tabla como antes se hacía en PAW, puedes simplemente escribir cosas como esto en los procesos, y DAAD Ready ya se encarga de colocar ese texto en un mensaje de manera que no tengas que hacerlo tú mismo:

MESSAGE "¡Hola mundo!"

Incluso tendrá en cuenta si pones dos mensajes idénticos, para usar sola mente una entrada de la tabla.

Aun así, la tabla puede ser usada cuando necesites mensajes con un número específico, para ser usado con indirección (por ejemplo, MESSAGE @100, escribirá el mensaje cuyo número esté en el flag 100).

La sección de Texto de Objetos (OTX)

Esta sección tiene un texto por cada objeto que haya en el juego, que es mostrado cuando el objeto es descrito. Un objeto es cualquier cosa en la Aventura que pueda ser manipulada o llevada, y se numera igualmente de 0 a 255. El objeto 0 es considerado por la librería que trae TEST.DSF como una fuente de luz.

La sección de Localidades (LTX)

Esta sección tiene una entrada por cada localidad, que será mostrada cuando lleguemos a cada localidad. Se numeran de 0 a 255, y la aventura empieza siempre en la localidad 0.

La sección de Conexiones (CON)

Esta sección tiene una entrada por cada localidad, que puede estar vacía o contener una o varios posibles movimientos.

Un movimiento se forma por un verbo (o nombre convertible) y una localidad de destino. Esto significa que, tecleando ese verbo desde la localidad indicada por la entrada, iríamos a la localidad de destino indicada en ese movimiento.

El contenido típico de una entrada podría ser:

```
SUR 6
ESTE 7
SALIR 6
NORTE 5
```

lo cual significa que SUR o SALIR o cualquiera de sus sinónimos llevarían a la localidad 6, ESTE o sus sinónimos a la 7 y NORTE o sus sinónimos a la 5.

Nota 1: Cuando se juega la Aventura solo el verbo de la SL genera movimiento.

Nota 2: Si un movimiento es llevado a cabo por una entrada en la tabla de respuestas que contenga una acción GOTO, puede no ser necesario que esté en la tabla de conexiones, a no ser que sea necesario para el movimiento de un PSI (PNJ) que pueda moverse incondicionalmente (ver condActo MOVE).

La sección de Definición de Objetos (OBJ)

Esta sección tiene una entrada para cada objeto que especifica:

- El número de objeto
- La localidad donde inicialmente está ese objeto.
- El peso del objeto.
- Si el objeto es un contenedor (por ejemplo, una maleta).
- Si el objeto es una prenda o algo que te puedes poner/quitar (por ejemplo, una gorra)
- Los atributos de usuario del objeto (ver HASAT/HASNAT).
- El nombre y opcionalmente el adjetivo asociados al objeto.

Nótese que el peso no está en una medida concreta, puedes usar kilos, gramos, o lo que quieras, pero el valor es en el rango 0-255. Simplemente valora después cuanto peso puede llevar el jugador usando ABILITY.

Un objeto puede ser perfectamente una prenda y un contenedor a la vez, por ejemplo, un bolso.

Las tablas de Procesos (PRO)

Esta sección es el corazón de las fuentes, generando la lógica del juego. Cada tabla consiste en varias entradas, cada entrada contiene un verbo y nombre para encajar con las órdenes del jugador, o bien el carácter "_" para indicar que el verbo, el nombre, o ambos, son indiferentes.

Al verbo y nombre sigue una serie de condActos, o bien otra entrada, en cuyo caso se considera que las entradas son equivalentes.

Por ejemplo, esto funcionará tanto si escribes ENCEDER LUZ como si escribes PULSAR interruptor:

- ENCENDER LUZ
- PULSAR INTERRUPTOR
  AT fGaraje
  ZERO fLuzEncendida
  MESSAGE "Enciendes la luz"
  SET fLuzEncendida
  DONE

Tabla de procesos 0

Esta tabla contiene el control principal de DAAD, se entra tras inicializar con la SL actual vacía. Normalmente consistirá en varias entradas '>_ _' y algún tipo de bucle.

Proceso 1 (y siguientes)

Son opcionales y definen sub-procesos que se llaman con el PROCESO 0.

Procesos que vienen con DAAD Ready

Nótese que el fichero TEST.DSF viene ya con una serie de procesos que más o menos simulan el comportamiento de PAW. Así, los procesos 0,1, 2 y 6 son internos para similar ese comportamiento, pero el proceso 3 es como el 1 de PAW, y el 4 es como el 2. Así mismo, el proceso 5 cumple la función de la tabla de respuestas.

Escape chars

Cuando se imprimen mensajes, puedes usar algunos caracteres espaciales que no se mostrarán como se ven, sino que harán algunos efectos en el texto.

Escape chars	Descripción
_ (guion bajo)	Es reemplazado por el objeto actualmente referenciado. Si se encuentra un artículo indefinido, lo reemplaza por el definido("un"->"el").
@	Igual pero el artículo reemplazado pasa a tener su primer letra en mayúsculas ("un"-"El"). Solo funciona en el intérprete español.
#b o #s	Imprime un espacio en blanco
#k	Cuanto este carácter es imprimido, el juego hace una pause y espera que pulsen una tecla.
#n or \n	Genera un salto de línea
#g	No imprime nada, pero a partir de ese momento los textos se pintarán con los 128 caracteres superiores del juego de caracteres (ver apéndices). Así, "#gt" pintará el carácter "t" del set superior.
#t	Vuelve a usar el juego de caracteres inferior.
#e	Escribe el signo del euro (€).

Comandos del preprocesador

Por favor, ten en cuenta que esta sección no es para novatos, así que no te preocupes si no la entiendes demasiado. De hecho, si eres nuevo, una vez leas #define e #ifdef puedes saltar al siguiente capítulo.

Todos los comandos del preprocesador empiezan por una almohadilla ('#').

#define symbol expression

Define la etiqueta "symbol" para que tenga el valor dado por la expresión. Puede ser un número, otros símbolos, o incluso una operación matemática entre comillas. Los símbolos pueden en general

ser usados allá donde se puede usar un número. Esto no afecta a los #ifdef que veremos más adelante, que aceptan un solo símbolo.

```
#define horas 24
#define dias 2
#define horasFinDeSemana "days*hours"
#define Mediodia "hours/2"
```

Ten en cuenta que el compilador es de un solo paso por lo que debes hacer los #define antes de usar los símbolos.

Una vez que símbolo se ha definido no puede definirse de nuevo, para eso usa #var.

El compilador define de manera automática varios símbolos. Leas asigna un valor 1 dependiendo de la máquina objetivo:

"pc","zx","cpc","msx","msx2","c64","cp4",etc.

Además, también define símbolos para sub targets, cuando lo hay, por ejemplo MODE_plus3 o MODE_next" para Spectrum

Además, genera los símbolos BIT8 o BIT16 dependiendo del target.

Así mismo, crea dos símbolos: ROWS y COLS cuyos valores son las filas y columnas de texto que hay en esa máquina

#define existe por dos razones:

1) Ser posible referenciar cosas por nombre en lugar de número. Por ejemplo, si el objeto 2 es una lámpara, puedes hacer un #define y después referenciarla así:

```
#define oLampara 2

CARRIED oLampara
DESTROY oLampara
```

En lugar de la aproximación antigua:

```
CARRIED 2
DESTROY 2
```

Esto incrementa la legibilidad del Código cuando vuelves a revisarlo meses o años más tarde.

2) Para usarlo con "ifdef" (ver debajo).

#ifdef "symbol"
{#else}
#endif

Este grupo de comandos ocurren juntos. Poner #else es opcional. Cualquier línea que siga el #ifdef hasta el #else si lo hay o hasta el #endif se incluirán en el juego solo si el "symbol" tiene un valor que no es cero.

Si usas #ifndef en vez de #ifdef, ocurrirá al contrario.

```
#ifdef "zx"
MESSAGE "Es difícil hacer eso con un Spectrum"
#else
MESSAGE "Es difícil hacer eso sin un Spectrum"
#endif
```

Si añades esto al Código, cuando se compilara para "ZX" (cualquier tipo de Spectrum) el jugador verá el primer mensaje, en los otros target verá el otro.

Aunque eso es un ejemplo muy simple, puedes usar esto para hacer código que entre en el juego dependiendo de qué símbolos se han creado.

#include "filespec"

Hará que el compilador pase a leer de ese otro fichero ("filespec") hasta que finalice, momento en el que seguirá leyendo en el DSF actual.

No puedes poner un include en un fichero que ha sido a su vez incluido.

#echo "text"

Muestra el texto en consola. Generalmente se usa para tener feedback visual de si un #ifdef ha ocurrido o no.

Ejemplo:

```
#ifdef "PC"
#echo "Incluyendo el fichero de cosas particulares para PC"
#include \LIB\PCDISP.DSF
#endif
```

#incbin "filespec"

Esto incluye un fichero binario tal cual viene en la DDB. No es algo

muy común y está relacionado con asuntos técnicos de los EXTERN que son cosas un poco complicadas para el alcance de DAAD Ready.

#defb "expression"

Similar a INCBIN pero directamente le damos un valor de byte en la expresión que incluir.

```
 #DEFB 1
 #defb "PSIFLAG-1"
```

#defw "expression"

Similar a DEFB pero incluye un Word, es decir, dos bytes.

```
 #defw 6578
 #defw IOADDR+4
```

#defb/w se pueden usar preferentemente a INCBIN si quieres incluir solo unos pocos bytes y dependen de otros símbolos.

#dbaddr symbol

Le da al símbolo el valor actual de la dirección en la base de datos. Se puede usar luego para CALL, #userptr o #defw.

#userprt n

Donde n es un valor de 0 a 9.

Este comando es para evitar el problema de las referencias forward en un compilador de un solo paso. Pone la dirección actual en uno puntero, en la tabla de punteros que hay en un sitio fijo al principio de la DDB, así una rutina externa puede ir a esa tabla a buscar el puntero.

#extern "filespec"
#sfx "filespec"
#gfx "filespec"

Solo para 8 bit.

Estos tres comandos son similares a #USERPTR, pero los valores son copiados en una tabla que luego es usado por los condActos EXTERN, SFX y GFX para el salto. Nótese que SFX ya tiene un valor por defecto para escribir en registros del chip de sonido, así que cualquier rutina que use esto reemplazará eso.

#int "filespec"

Solo para 8 bit. Cualquier rutina puesta en ese binario será llamada 50 veces por segundo durante el juego. No hay por tanto comando en DAAD para llamar a esta rutina. En Z80 se preservan solo los registros AF y HL (dado que HL es dado como tu address), así que asegúrate de preservar los registros que uses. En los intérpretes 6502 se preserva todo el estado del procesador.

Al incluir el nombre de fichero en un #EXTERN, #SFX o @INT el fichero se incluirá en la base de datos DDB como si se hubiera hecho con INCBIN, pero además actualiza los punteros internos.

Apéndices

A – El juego de caracteres

El Sistema de DAAD usa un juego de caracteres de 256 caracteres.

DAAD Ready está usando algunos caracteres del bloque de 128 caracteres superiores para representa caracteres internacionales que no estuvieran ya soportados antes (los españoles.

DAAD Ready permite cambiar el juego de caracteres de tu aventura:

Si quieres cambiarlo, tienes que modificar los ficheros que hay en la carpeta ASSETS/CHARSET. Estos serían:

- AD8x6.CHR es por Spectrum, MSX1, MSX2, Amstrad PCW y PC.
- AD8x8.CHR es usado por Amstrad CPC.
- C64bold.CHR es usado por C64 y Plus/4.

Para modificar el Fuente, puedes usar la aplicación GCS que está en TOOLS\GCS. ZX-Paintbursh también puede abrirlos. Si grabas la fuente con GCS, grábalo para 8 bit, incluso si vas a usarlo para un sistema de 16 bit. DAAD Ready se encarga de eso.

Por favor ten esto en cuenta:

- AD8x6.CHR se muestra como un Fuente de 6 pixeles de ancho, así que debes evitar usar las tres columnas de pixeles de la derecha. Las dos más de la derecha ni siquiera se mostrarán, la tercera, si la usas, las letras quedarían pegadas unas con otras.
- AD8x8.CHR es una fuente de 8 pixeles de ancho. A pesar de ello está definido como una fuente de 6 pixeles, que es por lo que en CPC el espacio entre letras es mayor. Puedes redefinir esta fuente para usar anchos reales, y así utilizar 7 columnas (dejando la octava para que haya espacio entre letras).
- C64bold.CHR es una Fuente en negrita, porque en TVs CRT conectadas al C64 vía antena, las letras finas son poco legibles.

Por favor ten en cuenta que si redefines los caracteres usados para los idiomas internacionales no podrás usarlo para esos caracteres. Si vas a hacerlo porque necesitas poner cualquier otro carácter o gráfico ahí, siempre puedes hacerlo empezando por los que pertenecen a un idioma que no estés usando. Por ejemplo
"ñ" si tu juego es en alemán, o "õ" si es en español).

B – Finalizando juegos de Spectrum Next

Hay un pequeño problema al finalizar (el jugador dice que no quiere echar otra partida) en Spectrum Next. El problema es que es imposible para el condActo END de DAAD saber que el gráfico está en el layer2, por lo que no lo borra, y un reset del Spectrum Next tampoco lo hace. Como resultado el gráfico se queda ahí, aunque el ordenador se resetee, aunque por supuesto, apagarlo y encenderlo restaura la situación normal. Aun así, como queda un poco raro, esta es una solución:

```
#define yesScancode 115;"s" keyboard code, para English es 115
#define noScancode 110;"n" keyboard ncode

   > _     _       SYSMESS 13; ¿Jugar de nuevo?
                   NEWLINE
                   NEWLINE
                   PAUSE 10

   > _     _       INKEY
                   SKIP 1; Se pulsó algo, veamos qué


   > _     _       SKIP -2; Nada pulsado, esperar


   > _     _       EQ 60 yesScancode
                   GOTO 0
                   RESTART

   > _     _       EQ 60 noScancode
                   XNEXTCLS
                   XNEXTRST        ; Limpieza del Next y reset

   > _     _       SKIP -5; no se pulsó ni N ni S
```

A partir de ahí, allá donde pondrías el condActo "END" pones una llamada a un "PROCESS x", donde has puesto el código arriba indicado.

Esto no se ha incluido en TEST.DSF porque es demasiado específico para Spectrum Next, y es difícil determinar dónde usarás el condActo END.

C – Personalización de DAAD Ready

DAAD Ready guarda algunos ajustes en el fichero CONFIG.BAT, que Es cargado por todos los otros .BAT. Además, DAAD Ready comprueba si existe el fichero CUSTOM.BAT. Y si existe lo carga después de CONFIG.BAT.

Eso permite a herramientas de terceros como Adventuron, integrarse con DAAD Ready e incluso cambiar cosas internamente.

D - Agradecimientos

- Tim Gilberts, de Gilsfot/Infinite Imaginations, por crear DAAD
- Andrés Samudio, de Aventuras AD, por permitir distribución gratuita de DAAD
- Richard Wilson, por WinAPE
- César Hernández, por ZEsarUX
- Mochilote, por CPCDiskXP
- Attila Grósz, por Yape
- Natalia Pujol, por el nuevo intérprete MSX2
- Imre Szell, por el nuevo intérprete Plus/4
- Marcin Skoczylas, por C64Debugger
- Habi, por CP/M Box, PCW Emulator
- A todos aquellos trabajando en dosbox, VICE64, OpenMSX, etc.
- A Javier San José, por el editor de fonts GCS
- A John Newbigin, por el DD para Windows
- A Amstrad y Locomotive, por permitir el uso de sus ROMS en los emuladores de Spectrum y Amstrad.
- A Juan José Torres, por el logo de DAAD.
- A Chris Ainsley, por sus ideas, y por el plugin de Visual Studio Code.

E -Licencias

DAAD Ready contiene software de un montón de partes, y ha sido creado por Uto (@utodev) que es el autor de Maluva (la extensión de DAAD) y DRC (el nuevo compilador). Todo el software usado es free o de código abierto, y en algunos casos donde la licencia no estaba especificada, el permiso ha sido facilitado por el autor.

- C64DEbugger es (C) Marcin Skoczylas, aunque usa parte del Código de Vice64. Si te gusta considera hacer una donación de cerveza a slajerek@gmail.com, o si prefieres dinero, directamente a su Paypal: http://tinyurl.com/C64Debugger-PayPal
- Yape es (C) Attila Grósz y es freeware. Mas información en http://yape.plus4.net
- DOSBOX es (C) Peter "Qbix" Veenstra, Sjoerd "Harekiet" van der Berg, Tommy "fanskapet" Frössman, Ulf "Finster" Wohlers. Encuentra más información y código Fuente en dosbox.com
- CP/M Box (PCW emulator) es (C) Habisoft. Mas información en http://www.habisoft.com/pcw/index_uk.asp
- MSX2DAAD is (C) NataliaPC, más información y código fuente en https://github.com/nataliapc/msx2daad/wiki.
- CPCDiskXP es (C) Mochilote. Más información en http://www.cpcmania.com/news.htm
- C1541 es parte de Vice64. Más información en https://vice-emu.sourceforge.io/
- dsktool es (C) Ricardo Bittencourt, Tony Cruise and NataliaPC. Fuentes e información en https://github.com/nataliapc/MSX_devs/tree/master/dsktool
- PHP es (C) varios autores. Mas información en php.net
- DD para windows es (c) John Newbigin y está licenciado bajo la GPL v2
- GCS para DAAD es (C) Javier San José and Uto, y su licencia está pendiente de definir
- Un número de imágenes de ROM son facilitadas con WinAPE y ZEsarUX, que son (C) Amstrad Plc y Locomotive Software. Amstrad y Locomotive han dado permiso para el uso de estas ROMs con emuladores, pero retienen el copyright.