PTCYR#10

Pliego de tonterías retro. Segunda época.

Panfleto Extendido

En esta segunda época incluimos un pliego central extra que podéis separar y coleccionar aparte. ¡Dos panfletos por el precio de uno!

Kiere pelea?

Tras dieciocho años de silencio volvemos más viejos, más cascarrabias pero menos gañanes (o eso esperamos) con el panfletillo de tonterías de la *retrowhatdefuk*. Lo que os presentamos puede llegar a ser muchas cosas, aún no tenemos claro qué, pero por lo menos intentaremos que sea todo verdad y que resulte más o menos divertido de leer y tal. Ojal, nos miraremos mucho el ombligo, que para eso es nuestro y está limpio.

Holi

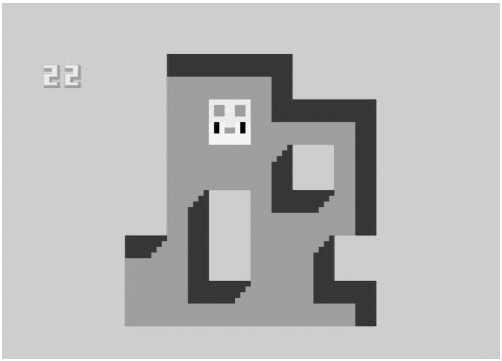
Illo, pasando de ponerme ahora con la típica parrafada de "hemos vuelto mu contento con papa y con pimiento". Eso es obvio. Empezamos que hay poco sitio y yo escribo mucho.

Güegos que he gugao

Estos últimos días he catado algunas producciones retro de los últimos meses y hay unas cuantas que me han gustado y que no sé por qué no había visto (mentira: sí lo sé: es que no echo cuenta). En el número de hoy os traigo cosas de *Astra Pestecé*. En el próximo pondremos de *ZX Peste a Rectum*.

Longato

Es un juego de puzzles muy chulo y maravilloso donde hay que cubrir el espacio libre con tu gato largo. Como los gatos molan ya tenían la mitad del trabajo hecho, pero es que además es bonito y se mueve guay.

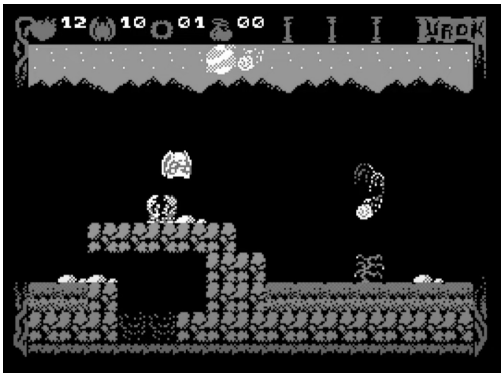


El güego es de gratis y además te dejan jugarlo online para que ni siquiera ya tengas que bajarte el archivo (qué pena, ¿en qué se está convirtiendo este mundo?), pero yo me lo he bajado porque ahora soy más viejo, más sabio y más cascarrabias (otra vez). Pruébalo, que además te dejan tocarle el totete (código fuente disponible).

►*https://tinyurl.com/y2ep6pdz*

Urok

Este está hecho con la *Churrera* y me encanta cuando hay ampliaciones y cosas nuevas. Urok es un plataformas divertido que lleva cosas *custom* en el movimiento del personaje y tipos de enemigos nuevos. Aunque los gráficos son sencillos me han gustado mucho y el cambio de aspecto de las zonas internas y las externas está muy bien logrado.



Igual podrían haberle puesto máscaras más grandes a los *sprites* (no, no

Sigue en la página 7 ›

¡EXCLUSIVA! Se está preparando un concurso de videojuegos en el que se premie al videojuego más tiralevitas, al videojuego con más curas, y al videojuego que esté más hecho en ensamble de todos. Más información PROXIMAMENTE.

MK1v3.2 para sanar padrastros

Haciendo cosas chulas para depurar MSC4

Ya lo hablaremos largo y tendido en otro sitio, pero asín rápido: MK1v3.2 es una versión 100% escrita en ensamble de la última versión de la Churrera clásica tal y como estaba en 2011, y que he estado utilizando para "cosas" y también para integrar y depurar el nuevo compilador e intérprete de scripts churreros, la versión 4 de MSC.

Desde que volvimos a desarrollar MK1 en 2020, en los juegos en los que yo me he encargado de programar, he preferido usar inyección de código a usar el sistema de scripting que, por otro lado, lleva sin evolucionar apenas desde 2015. El verano pasado se me ocurrieron unas cuantas cosas (eso aún no te lo puedo decir!!) para las que vendría bien el sistema de *scripting*, pero tal y como estaba el tema era pesado y necesitaba una limpia en cuanto a la sintaxis y cómo se hacían las cosas (se nota que estaba hecho a bocaos y empellones).

Ponerse a tocar el mostrenco no me parecía buena idea, con código que llevaba ahí desde 2010. El tema de las cláusulas (secuencias de comandos que se ejecutan si se cumple una lista ordenada de comprobaciones) me seguía pareciendo la mejor forma de montar un scripting para un motor de acción de 8 bits. El problema del viejo MSC es que lleva un montón de comprobaciones distintas y un montón de comandos distintos que tienen que ver con diferentes partes del motor a las que hace falta acceder. Cada comando o comprobación lleva código para interpretarlo. Cuando más complejo sea el *script* (cuantas más acciones o comprobaciones distintas use) más tocho es el intérprete. Luego pasa que además es un poco lentete. Nada, esto hay que hacerlo de nuevo.

La nueva forma de enfocar el tema es a través de un evaluador de expresiones mucho más potente, que permita comparar o asignar *flags* o valores, o incluso variables del motor, de forma totalmente transparente. En el MSC viejo teníamos una comprobación para ver si un *flag* valía tal valor, otra para comparar que un *flag* era igual a otro *flag*, otra para comprobar que el número de objetos era tal valor, otra para comprobar que la posición x del jugador era tal... Con el nuevo sistema sólo tendremos una comprobación: $N = M$, en la que N y M pueden ser *flags*, valores, el número de objetos actual y muchas cosas más. Y además en ensamble.

MSC4, tanto compilador como el intérprete (que se genera de forma dinámica según el *script* que compilemos), son mostrencos importantes. Una de las razones para montar MK1v3.2 fue la de tener un motor rápido, limpio y *bare bones* para probar. La MK1v3.1 de 2011 fue la primera versión de la churrera que consideramos completa. Tomando el motor en ese punto de la historia y rescribiéndolo usando los algoritmos de las versiones actuales y en ensamble nos construimos el mejor banco de pruebas posible para MSC4.

Ya hay pruebas y cosas hechas a mansalva (pero aún no te las puedo decir!!), pero vamos a usar estos coleccionables para hacer más, y aprovecharemos para *sanar padrastros*: recuperaré ideas que hemos tenido desde que la Churrera es Churrera pero que se quedaron en el tintero por una razón u otra. Y, para empezar, nos ocuparemos de una historia que se le ocurrió a Anjuel al principio de los tiempos, con la Churrera apenas estrenando la ver-

sión 2.0...

Khe

En el internec tienes dos tutoriales de la churrera y el viejo manual. A estas alturas del cuento se entiende que una intro sobre cómo funcionan las cosas no es necesaria. ¡Esto es para iniciados!

COSMONAUTIC

Como decíamos, esto vino de una idea de Anjuel allá por 2010, justo cuando estábamos metiéndole la vista genital a la churrera. El tema iba de un astronauta que iba por ahí y tal y cual y hacía cosas. Por aquellos entonces no había motor de Jetpac, pero se nos ocurrió algo muy chulo para simular la atmósfera cero: ¡la vista lateral falsa! Ejercitando nuestra mundialmente conocida técnica de *engañar al chamán*, pensamos que configurando el motor para vista genital pero poniendo los gráficos como si fuera vista lateral y subiendo la inercia a tope daría todo el pego. Las teclas de dirección Q O P A simularían "empujes" de pequeños propulsores en el traje del astronauta y molaría todo que te cagas. Al menos en nuestra cabeza era increíble pero... ¡nunca llegamos a probarlo!

Quince años más tarde navegaba por el foro de mojonia buscando ideas y digo mira, es el momento de probarlo. El modo *FAKE_SIDE_VIEW* fue añadido a MK1v3.2 que hice sin ningún tipo de vergüenza ni remordimiento porque no añadía código nuevo, sino que simplemente activa algunas cosas de la vista genital y otras de la vista lateral. De hecho, este modo es vista genital 100% pero la selección de *facing* es como la vista lateral: embéz de haber 4 direcciones en las que el *sprite* puede mirar, sólo hay dos que se activan con izquierda o derecha. Pulsar arriba o abajo no modifica adonde mira el muñeco.

Sí, a ver, es que esto en 2010 no se nos ocurrió cuando estábamos ahí flipando con el marciano en nuestra imaginación. Pero era lo único que no estaba en el motor, LO HÚRO.

Además del modo falso, hay que activar *PLAYER_CUSTOM_FRAME* para que la selección de qué *cell* de animación se pone en cada momento la hagamos nosotros desde *custom.h*. Y la hacemos tal y como se muestra en el cuadrito, recordando que MK1v3.2 pone qué dirección estamos pulsando en la variable *thrusting* o un 0 si no estamos pulsando nada:

```
unsigned char player_custom_frame (void) {
  // Return a frame NUMBER (0-7)
  #asm
      ld  c, 0

      ld  a, (_thrusting)
      or  a
      jr  z, m_frame_set

      inc c
      cp  THRUST_UP
      jr  z, m_frame_set

      inc c
      cp  THRUST_DOWN
      jr  z, m_frame_set

      inc c

    .m_frame_set
      ;; player.facing
      ld  a, (_player + 22)
      add c
      ld  h, 0
      ld  l, a
      ret
  #endasm
}
```

Lo que hace lo de arriba es cambiar el

frame dependiendo si estamos pulsando teclas. Recordemos que estamos en genital, pero el *spriteset* es este:

O sea, sin propulsar, propulsando hacia arriba, hacia abajo, y hacia el

frente. El código simplemente va incrementando el número de *frame* mientras va comprobando. Si estamos pulsando arriba se quedará en el 1, si pulsamos hacia abajo se quedará en el 2 y en caso contrario en el 3.

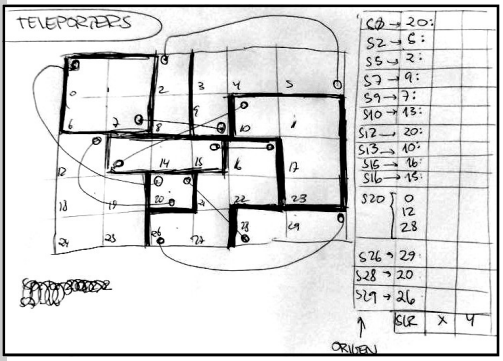
Además de eso, tenemos unos valores de inercia que seguro que Chema Velo adorará:

```
#define PLAYER_MAX_VX   256
#define PLAYER_AX       8
#define PLAYER_RX       1
```

¡Y funciona! Ya en 2010 (mayormente) estábamos en lo cierto: esto iba a dar el pego y *engañar al chamán* seguiría siendo la mejor forma de trabajar! Pero eso es lo de menos, había más planes ¡más planes! ¡Estábamos loquísimos en 2010!

Teletransportadores

¡Sí! El mapa no sería lineal, sino que estaría dividido en secciones disjuntas



de varias pantallas con formas irregulares que se pegarían como un Tetris y que estarían conectadas con teletransportadores.

En aquel momento de la historia MSC estaba apenas pensao y las ideas eran muy poco ambiciosas, así que la forma que se nos ocurrió hacerlo era ampliando el motor para implementar una lista de teletransportadores que bla, bla, bla.

Pero recordemos que lo que queremos en este momento de nuestras vidas es "hacerlo posible con MK1v3.2 tal cual tirando del nuevo MSC4". Sí, vamos a hacer esto por *scripting*. Y vamos a ver cómo ¡no va a impactar en nada y además va a ocupar poquísimo!. Y de regalo podremos usarlo para otras cosas.

¿Qué nos ofrece MK1?

Por un lado tenemos el comando *WARP TO*, que permite mandar al jugador a otra pantalla y coordenadas. ¿Para qué vamos a implementar un sistema de teletransportadores si podemos hacerlo fácil con MSC4?

Dejaremos que el jugador se acerque al teletransportador, que es el tile 19, y pulse FIRE. Como tenemos el *scripting* activado y que se lance con *FIRE*, esto ejecutará la sección *PRESS FIRE AT ANY* y luego *PRESS FIRE AT n*, con n la pantalla actual. Pero es que eso no nos hace falta, podemos comprobarlo todo en *PRESS FIRE AT ANY*. ¡El intérprete es ahora tan rápido que aunque fallen todas las comprobaciones (por ejemplo si pulsamos FIRE donde nos de la gana) no habrá ningún tirón, como pasaba en la *Churrera de antaño*!

```
NOINDEXED
FASTNPANT

PRESS FIRE AT ANY
  IF TN != 19
  THEN
    BREAK
  END

  IF TRUE
  THEN
    EXTERN 0    # kjjj
  END

  IF NPANT = 2
  THEN
    WARP TO 5, 10, 8
    BREAK
  END

  IF NPANT = 5
  THEN
    WARP TO 2, 2, 3
    BREAK
```

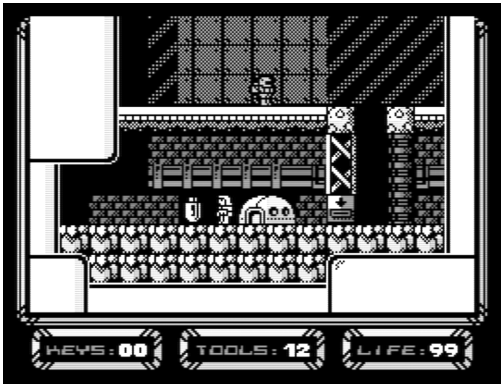
```
  END

  # Etcétera…

END
```

Fijaos como ahora todas las comprobaciones son menos ortopédicas ¡todo está integrado! La variable *TN* contiene siempre el número de tile que estamos tocando, y *TX* y *TY* serán sus coordenadas, por si necesitamos tener más de un teletransportador en la misma pantalla. *NPANT* contiene la pantalla actual.

El *script* empieza con un par de directivas. La primera, *NOINDEXED* indica que no habrá secciones específicas para cada pantalla. Así el compilador no generará un índice y ahorraremos de la leche, especialmente si hay muchas pantallas. La segunda, *FASTNPANT*, genera un *bytecode* especial para *IF NPANT = X* (de 2 bytes) y un intérprete específico en vez de tratar



NPANT como variable normal. Si hay muchas comprobaciones de *NPANT* (como es el caso) esto es a mejor opción.

PRESS FIRE AT ANY se ejecuta siempre que pulsemos FIRE. Lo primero que hace es comprobar si el *tile* que tocamos es 19. Si no lo es, abortamos del tirón.

Si estamos tocando el *tile* 19, ejecutará el *EXTERN 0* que es código ensamble que hemos añadido en *extern.h* a la vieja usanza y que hace un efecto de sonido y visual. Y luego seguirá interpretando la siguiente cláusula.

Para cada pantalla simplemente mandamos a otra. *BREAK* hace que no se ejecute más código. Faltarían más pantallas pero esta es la idea.

Pinchos *custom*

El comportamiento estándar genital de chocarse con pinchos no nos vale para este juego. El choque pinchil siempre ha sido un problema en la churrera ¡nunca dábamos con la solución buena! Es por eso que MK1v3.2 permite pasar del que trae y hacer el tuyo, con casinos y furcias. En este caso necesitamos uno que de preferencia a *vy* sobre *vx*. Para eso se activa *PLAYER_CUSTOM_BG_HIT* en el *config.h* y se hace uno su propia función en *custom.h* como mejor venga. Ya veréis la rutina cuando publiquemos el juego y las fuentes, don't worry, es que aquí ya va cabiendo poco más.

Nunca usado nunca

Este güego trae *BOTI_BOINS*, que son tiles que te rebotan. Eso estuvo un tiempo en la churrera pero desapareció porque me equivoqué al ir pasando

En el próximo número sacaremos mucho más la picha con MSC4 poniendo una lista de las comprobaciones, comandos y variables que se pueden usar, y a lo mejor ponemos ejemplos reales de juegos de verdaz y os contamos alguna mierda que se nos ocurra para ilustraros y entreteneros. Y diréis ¿cómo consigo esa maravillosa MK1v3.2? Pues pronto la pondré para bajar en algún sitio de esos de bajar o en un sitio de donde los frikis.

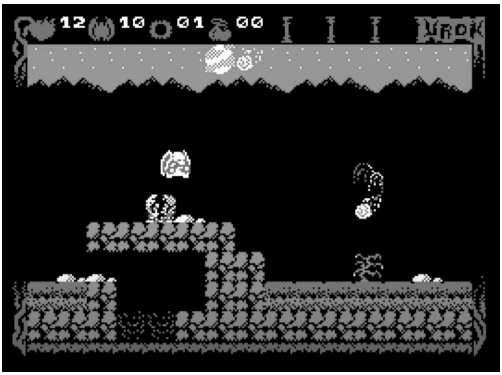
‹ *Viene de la página 2*

son Batman) para darles un reborde que hiciera más fácil separarlos del escenario, ya que a veces se mezclan cuando tienen el mismo color (vamos, que se dejan de distinguir bien, nada tan grave como que se conviertan en una explosión de *glitch XOR* multicolor cuasi psicotrópica como ocurre en otros motores). En definitiva se trata de un juego divertido que le sabe sacar partido el motor..

►*https://tinyurl.com/3j2ejkah*

Tony Montezuma's Gold

Lo comentaba el otro día con los mojones, que siempre que veo un juego monocromo en CPC pienso que igual hubiera molado hacerlo en modo 2 en vez de modo 1. Algún día escribiré las rutinas que faltan en *CPCRSLIB* y haremos uno los Mojón Twins. Uno que sea tétrico y oscuro, con gente rara y desagradable y pantanos llenos de plantas pie y arbustos de postillas.



El Montezuma's este es monocromo en modo 1 (he jugado en CPC, otro día jugaré a la versión marrón, o no), tiene buen diseño y se juega bien. El movimiento es un poco brusco a veces (en cuanto al *sprite* desplazándose de n en n *pixels*) aunque esté muy bien animado. Hay saltos, agachaciones y subidas y bajadas por las escaleras, como le gusta a mi amigo Anjuel. A veces también rasca y parpadea un poco pero bueno, con la presbicia se nota menos. Yo he jugado a la demo y no sé si ya ha salido la versión completa o qué, lo siento, vaya mierda de panfleto que el que lo escribe un sentera un caráo. El juego lleva un tebeo de presentación asociado, cosa que siempre gusta. Podrían haber hecho la portada del mismo estilo que el tebeo en vez de usar una mierda de *IA robaimagenes secaacuíferos*, igual así el personaje no tendría una correa misteriosa que se le mete por el bolsillo de la camisa, pero bueno. Dale argo a los *devs* en Pichío aunque no quieran gastar en diseñadores de carne hueso, no me seas rata.

►*https://tinyurl.com/mr2d2rfe*

Entrevista a Anjuel

Para esta nueva segunda época de este panfleto vamos a entrevistar a los Mojón Twins en el orden en el que me aparecen en el Telegram. Por tanto empezamos por Anjuel.

PTCYR ¿Cómo se hace un juego de MK1 para que le pongan un 90%?

Anjuel: Me congratula profundamente que me plantees semejante cuestión. En realidad, alcanzar semejante pun-

tuación no depende tanto del código como del aura conceptual que impregna cada línea del motor. Sin embargo, conviene recordar que la auténtica sofisticación reside en la impostura: basta con colocar un logo de AGD —esa herramienta de masas tan plebeya en su accesibilidad— para sugerir que se ha alcanzable una pureza lúdica inalcanzable para el común de los mortales. El MK1 es para los que comprenden, el AGD es para los que apenas intuyen.

P: ¿Tienes algún proyecto en mente? ¿Kiere pelea?

A: Mis ideas se amontonan como libros prohibidos en una biblioteca que arde lentamente bajo el peso del tiempo. Me pone burrísimo la idea de resucitar el espíritu de Super Yum Yum, esa joya oculta de la era pre-smartphone. Me encantaría adaptarlo en CPC con su gran gama de colores, no como un mero homenaje, sino como una reinterpretación filosófica del acto de comer frutas digitales en un mundo que ha olvidado el sabor del píxel. Kiero pelea.

P: Hace tiempo que no vas a Cepeceros, ¿qué güego llevarías?

A: Sin duda optaría por un hallazgo que destilase tanto elegancia como arrojo arqueológico, algo del calibre de The Shadow of the Sword, un Shinobi la mar de apañado. Y, para equilibrar el tema, llevaría también una pieza infame, una catástrofe lúdica que desafiara las nociones mismas de diversión... por ejemplo, Tuma 7, una experiencia que trasciende lo deleznable para situarse en la esfera de lo metafísicamente injugable solo al alcance del gran creador de Ulises. Y nada mejor que torturar a mi amigo Chemica, quien tendría que sufrir jugándolo, en un acto de justicia poética y sadismo retro.

P: ¿Nicanor para cuándo? ¡Versión Commodore ya!

*A:*Nicanor... ese padrastro mítico que me persigue hasta en la sopa. La ¡versión para Commodore ya! es, sin duda, una deuda moral, un acto de reparación histórica con la máquina que nunca pidió disculpas por sus limitaciones y que tan buenas tostadas con mantequilla hace. Me gustaría pensar que algún día, impulsado por el orgullo y mi resentimiento con este tema, encontraré las ganas necesarias para seguir adelante. Hasta entonces, me consuela saber que el mito crece cada vez que alguien pronuncia su nombre en vano o alguien lee el número 0 de Monomanía.

P: ¡Gracias, señor Anjuel!

Pos ya estaría

Esto es lo que ha dado de sí el pliego. A ver qué tal sale el experimento del coleccionable, que me ha salido más denso que un consomé de alquitrán. El número que viene estará si eso ya, y traerá un montón de cosas, pero eso aún no te lo puedo decir!!!