Manual del PIX37

Editor, després codi. Còpia fora de línia del llibre a pix37.party/docs.

Editor

PIX37 DEV és l'editor. Dibuixa, escriu i escolta el joc aquí, i després exporta un cartutx de joc o una compilació independent. També pot reproduir un cartutx de joc i pot obrir PIX37.PARTY per descarregar cartutxos de joc i utilitzar el catàleg. PIX37 DEV EDUCATIONAL és el mateix editor i és gratuït. No té compilacions, ni exportació de cartutxos de joc ni accés a PIX37.PARTY. PIX37 Console és el reproductor. Reprodueix cartutxos de joc `.p37` i obriu PIX37.PARTY per descarregar cartutxos de joc i navegar pel catàleg.

Projectes i cartutxos de joc

Un fitxer `.pix37` és el projecte: pestanyes font, fulls d'sprites, mapes, so i taules de dades. Un fitxer `.p37` és el cartutx de joc que executa el jugador: bytecode compactat.

Les partides desades i els assoliments es troben a la carpeta d'autor, al costat de les dades de l'aplicació. A Windows, això sembla `C:\\Users\\Ada\\AppData\\Roaming\\PIX37\\Mira\\Pong.p37s` per a una partida desada i `C:\\Users\\Ada\\AppData\\Roaming\\PIX37\\Mira\\Pong.ach` per als assoliments. A Linux, els mateixos fitxers són `~/.local/share/PIX37/Mira/Pong.p37s` i `~/.local/share/PIX37/Mira/Pong.ach`. Ada és el compte de l'ordinador. Mira és el nom de l'autor i aquest nom és la carpeta. Pong és la ranura per desar, que per defecte és el títol del joc. `save` escriu el `.p37s`. El primer assoliment desbloquejat escriu el `.ach`. Si els fitxers ja hi són, el runner els llegeix quan s'inicia el cartutx del joc.

Menú de desenvolupament

La tecla Esc obre el menú de desenvolupament. La tecla Esc mentre s'executa una partida torna a l'editor. Confirma amb Enter o el botó A (V per defecte). La tecla Enrere és el botó B (C per defecte).

Sense cap projecte carregat, el menú és: CREA PROJECTE, CARREGA PROJECTE, COMPILA, CARREGA cartutx de joc, CONNECTA A PIX37.PARTY, CONFIGURACIÓ.

Amb un projecte carregat és: DESA EL PROJECTE, CARREGA EL PROJECTE, CREA EL PROJECTE, ACTIUS, COMPILA, CARREGA EL cartutx de joc, CONNECTA A PIX37.PARTY, CONFIGURACIÓ.

Ctrl+S desa el projecte obert. Ctrl+R l'executa.

Crea un projecte

CREATE PROJECT demana el cartutx del joc abans de començar a crear el projecte.

- **VÍDEO** — 1 BIT (2 colors), 2 BIT (4), 4 BIT (16), 8 BIT (256).
- **ÀUDIO** — profunditat de mostra del sintetitzador: 16, 8, 4, 2 o 1 bit.
- **PANTALLA** — les configuracions predefinides inclouen 64×64, 128×128, 160×90, 256×256 i 320×180. La mida PERSONALITZADA us permet triar qualsevol mida d'1×1 a 512×512.
- **PALETES** — bancs de colors. El primer banc comença com el conjunt de 16 colors següent. LOAD PALETTE aplica un conjunt integrat: PICO-8 16, PICO-8 32, Game Boy, 1-bit, CGA, Sweetie-16 i una llista més llarga extreta de Lospec (Resurrect 64, Apollo, Endesga 32 i 64, CC-29, SLSO8, Pear36, Oil 6, Duel, AAP-64, Lost Century i altres). També podeu carregar un fitxer de paleta o carregar el vostre propi PNG i extreure la paleta d'aquesta imatge.



0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

```
lette — black, navy, purple, green, brown, gray, silver, white, red, orange, yellow, lime, blue, indigo, p
```

Índexs de color utilitzats per `cls`, `pset`, `spr` i la resta. Podeu tornar a pintar qualsevol índex a l'editor de paletes.

Els cinc panells

L'editor és GAME, PAINT, CODE, SOUND i DATA, en aquest ordre. El mapa es troba dins de PAINT. Els efectes especials i les cançons es troben dins de SOUND. Feu clic en un panell per treballar-hi. Quan CODE té el focus, SOUND i DATA ignoren les tecles de lletres, de manera que en escriure no es toca també el piano.



> Disseny complet, escrit `1:1:4:1:1`. Cada número indica la part de la fila que ocupa el panell: GAME, PAINT, CODE, SOUND, DATA. Una pantalla petita es troba centrada en una cel·la de 128 × 128. Una pantalla més gran de 128 px eixampla la columna GAME.

CONFIGURACIÓ → GLOBAL → DISSENY cicla els arranjaments:

Codi	El que veus
`1:1:4:1:1`	Tot. JOC i SO apilats, PINTURA i DADES apilats, CODI a la dreta.
`1:1:4:1:1 2X`	La mateixa disposició. El CODI es dibuixa amb el doble de densitat, de manera que hi cap el doble de text.
`1:0:1:0:1`	JOC, DADES, CODI.
`1:0:1:0:1 2X`	El mateix arranjament, amb el CODE més dens.
`0:1:1:1:1`	Sense vista prèvia del joc. PINTURA, CODI, SO, DADES.
`0:1:1:1:1 2X`	El mateix arranjament, amb el CODE més dens.
`0:1:4:1:2`	Estudi. CODI ample, una tira de DADES més alta, PINTURA i SO.
`0:1:4:1:2 2X`	El mateix arranjament, amb el CODE més dens.
`0:0:4:0:4`	Només CODI i DADES, tots dos grans.
`0:0:4:0:4 2X`	El mateix arranjament, amb el CODE més dens.
`0:1:4:1:0`	PINTURA, SO i un CODI ampli.
`0:1:4:1:0 2X`	El mateix arranjament, amb el CODE més dens.
`0:4:0:0:0`	Només PINTURA, 256×256, un píxel del full per un píxel del llenç.

Una fila `2X` manté els mateixos panells. El panell CODE es dibuixa amb el doble de resolució i es redueix a la mateixa cel·la, de manera que el text, les pestanyes i el marge inferior tenen la meitat d'alçada i el codi font cap el doble a la pantalla.

JOC

La vista prèvia del cartutx del joc en execució. Feu clic al panell JOC o premeu Ctrl+R per executar aquesta vista prèvia. CONFIGURACIÓ → PANEL DE JOC és on es configura el panell:

- **Previsualització** : color inactiu, continu o recàrrega automàtica quan editeu.
- **Juga en una finestra nova** : el joc s'obre al costat de l'editor.
- **Pantalla** : vista prèvia com a RGB888 o RGB565.
- **GAME 64** — una pantalla de 64×64 està centrada 1:1 a la cel·la o escalada 2× per omplir-la.
- **Shaders** — CRT i filtres del joc, separats dels filtres de l'editor.
- **Color** — fons del panell.

PINTURA

Fulls de sprites i, al costat del mapa del mateix panell, mapes de mosaic. Els fulls poden ser més grans que la pantalla. Les mides de previsualització dels sprites són 8, 16, 32, 64 i 128 (128 només quan el full té com a mínim 128 px). Desplaceu la roda del ratolí per sobre de la previsualització dels sprites per ampliar-la. Les mides del pinzell són 1, 2, 3 i 4.

Clau	Eina
B	Llapis
M.	Selecció rectangular
Q	Llaç
S	Segell
G	Cub
Jo	Comptagotes
V	Moure
L / Majúscules+L	Línia / corba
U / Majúscules+U	Rectangle buit / el·lipse buida
F / Majúscules+F	Rectangle omplert / el·lipse omplert
D	Tremolor
[ ]	Pinzell més petit/gran
Ctrl+Z / Ctrl+Y	Desfer / refer. Ctrl+Maj+Z també refereix.
Ctrl+C / Ctrl+X / Ctrl+V	Copiar, tallar, enganxar
Ctrl+A	Seleccioneu el llenç

El mapa utilitza la mateixa idea, amb menys eines: B llapis, M selecció rectangle, S segell, V moure i Ctrl+Z, Ctrl+Y, Ctrl+C, Ctrl+V. Mantingueu premuda la tecla Alt per ampliar la textura del mapa. La mida del pinzell al mapa es manté amb les eines de pintura.

Cada sprite té vuit bits de bandera. Paint els estableix al sprite; `fget` i `fset` els llegeixen al joc.

CONFIGURACIÓ → EDITOR DE PINTURA és on es configura aquest panell:

- **Paleta** — quin banc es dibuixa, a més de carregar i desar.
- **Previsualització** : RGB888 o RGB565.
- **Cursor de pinzell** : el punter mentre pintes.
- **Ajust de píxels** : com el pinzell s'ajusta als píxels.
- **Pinzell gran** : un pinzell gran s'enganxa des de la part superior esquerra o des de tota la seva petjada.
- **Pell de ceba** : un fantasma d'altres fotogrames i la direcció que mira aquest fantasma.
- **Zoom del mapa** : píxels reals o zoom lliure.
- **Colors** — els colors de la interfície d'usuari de la pintura.

CODI

Les pestanyes són fitxers separats dins del projecte. Feu clic al nom d'una pestanya per canviar-li el nom; premeu Intro per confirmar. Diverses pestanyes que defineixen `_init`, `_update` o `_draw` s'uneixen en ordre de tabulació: la primera pestanya s'executa primer i l'última última. El codi que no es troba dins d'un ganxo passa a formar part de `_init`. Un cartutx de joc sense cap ganxo es tracta com un `_draw`.

Clau	Acció
Fletxes	Mou el cursor. Prem Majúscules i estén la selecció.
Ctrl+Esquerra / Dreta	De paraula.
Inici / Fi, Ctrl+Inici / Fi	Inici o final de la línia o de la tabulació.
Pestanya	Inseriu dos espais. A la llista de finalització, accepteu el nom.
Introdueix	Nova línia, amb sagnat com l'anterior. Accepta una finalització.
Ctrl+C / X / V	Copiar, tallar, enganxar.
Ctrl+Z / Ctrl+Y	Desfer / refer.
Ctrl+A	Seleccioneu la pestanya.
Ctrl+D	Duplica la línia o la selecció cap avall.

Ctrl+/	Comenta o deixa de comentar amb `--`.
Ctrl+F	Cerca. F3 salta al següent cop. Esc tanca la barra.
Ctrl+G	Anar a un número de línia.
Escapada	Tanca primer la llista de cerca, l'opció "anar a" o la llista de pistes. Una segona tecla d'escapada surt de l'editor i obre el menú de desenvolupament.

Apareixen pistes mentre escriviu el nom d'una funció. Premeu Intro o Tabulador per inserir el nom i un `(` si COMPLET (està activat a la configuració. CONFIGURACIÓ → EDITOR DE CODI:

- **Cerca** — Ctrl+F a la pestanya activa o a totes les pestanyes.
- **Text alt** : retalla els glifs a 5 px o mostra el glif complet.
- **Mida de la cadena** : mida de la lletra dels literals de cadena.
- **Font de comentaris** : 3×5 o la font Fusion de l'idioma actual de la interfície d'usuari.
- **Complet (** — indica si la compleció escriu el parèntesi.
- **Suggeriments de codi** : activats o desactivats.
- **Idioma dels consells** : l'idioma del menú o anglès.
- **Espai del circumflex** : desplaçament addicional a les línies buides.
- **Canaleta** — números de línia.
- **Color** : fons, comentaris, pestanyes, números, operadors, paraules clau, cadenes, funcions del motor, les vostres funcions, variables, claudàtors, circumflex, selecció, línia d'error, avís.

SO

El panell té dues pestanyes. SFX és un patró: una sèrie curta de notes. SOUND és una cançó: col·loca aquests patrons en vuit canals. TIC és quants tics espera un pas. El rellotge és de 120 tics per segon, i un pas és una setzena nota, de manera que 15 tics són 120 pulsacions per minut. Un TIC més alt és un patró més lent. Dos patrons es mantenen al mateix temps quan un TIC divideix l'altre: 15 i 30 sí, 15 i 8 no.

El piano sona quan es selecciona una nota amb el ratolí. Les tecles blanques són ZXCVBNM (CDEFGAB). Les tecles negres són SDGHJ (C# D# F# G# A#). Passeu el cursor per sobre de qualsevol valor i escriviu un número, o passeu-hi el cursor i feu moure la roda. Els números de to del codi van de 0 (C-0) a 63 (D#5).

En un patró, LOOP i LEN són els mateixos dos números. OUT per sobre de 0 fa un bucle des d'IN fins a OUT, i el títol diu LOOP. OUT a 0 i IN per sobre de 0 és LEN: es reprodueixen aquestes notes des del principi, després el patró acaba i el títol diu LEN. Tots dos a 0 es reprodueixen a cada pas.

La quantitat de sintetitzador que podeu editar depèn de la profunditat de bits escollida per al cartutx del joc. CONFIGURACIÓ → EDITOR D'EFECTES SFX defineix aquesta profunditat, si una nota avança automàticament (desactivada, cadena de notes o cadena de camps), si Space reinicia la reproducció o alterna l'inici i l'aturada, i els colors de l'editor.

16 bits

Cada pas té una nota, un volum, una ona i un efecte. El volum és de 0 a 7. 0 és silenciós.

Ones:

- **0 triangle**
- **1 serra inclinada**
- **2 serra**
- **3 quadrats**
- **4 polsos** — un pols estret
- **5 orgues**
- **6 sorolls**
- **7 fasers**
- **8 PWM** — un quadrat l'amplada del qual es mou
- **9 polsos 25%**
- **10 sinus**
- **11 soroll metàl·lic**
- **Campana de les 12 FM**
- **13 arrencar**
- **14 superserra**
- **15 PWM (C64)**

Efectes sobre una sola nota:

- **0 cap**
- **1 diapositiva** : el to i el volum es llisquen fins a la nota següent en aquest pas
- **2 vibrato** — oscil·lacions de to
- **3 caiguda** — el to cau en silenci, la puntada de peu habitual
- **4 esvaïments**
- **5 esvaïments**
- **6 arpegis** — camina sobre aquesta nota i les tres següents, una cada 8 tics. A un TIC 8 o inferior, cada 4 tics.
- **7 arpegis ràpids** : les mateixes quatre notes, cada 4 tics o cada 2 quan el TIC és de 8 o inferior.

La capçalera del patró, a sobre de les notes:

- **TIC** — pulsacions per pas, de l'1 al 255. 15 són 120 pulsacions per minut. Escriviu el número o aneu-lo augmentant-lo gradualment.
- **LOOP IN / OUT** — amb OUT per sobre de 0, el patró fa un bucle d'IN a OUT. Deixeu OUT a 0 i el títol esdevé LEN: IN és el nombre de notes que es reprodueixen i 0 significa el patró sencer.
- **SOROLL** — tot el patró es converteix en soroll.
- **CRUSH** — una variant animada de cada onada.
- **DTUNE** — 0 apagat, 1 un cor lleuger, 2 un cor fort.

- **RVERB** — reverberació: 0 desactivada, 1 curta, 2 llarga.
- **DAMPN** — un filtre de pas baix: 0 desactivat, 1 lleuger, 2 fort.
- **LEVEL** — volum de tot el patró, de 0 a 7.
- **SWING** — barrejar els passos. Els passos senars esperen una mica més, els passos parells una mica menys.
- **PASSOS** — quants passos utilitza el patró: 32, 24, 16 o 12.

8 bits

Un pas té una nota i una ona. Les quatre ones són quadrada, triangular, de serra i de soroll. La capçalera té TIC, LOOP IN/OUT, FX i LEVEL. TIC és un dels següents: 1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 16, 20, 24, 30, 40, 60. FX és un efecte per a tot el patró: 0 cap, 1 vibrato, 2 arpegi, 3 esvaïment. LEVEL és de 0 a 3.

4 bits

Un pas és només una nota. La capçalera té TIC, LOOP (activat o desactivat), WAVE (les mateixes quatre ones) i OCT (en quina octava es troben les notes). TIC és un dels següents: 4, 6, 8, 10, 12, 15, 20, 30.

2 bits

Un pas és buit, un tambor, un cop o un aixafament. La capçalera té TIC i BASE. TIC és 8, 15, 16 o 30. BASE estableix l'alçada a la qual es troben el tambor i el cop.

1 bit

Un pas és una nota. TIC és 8, 15, 16 o 30, les mateixes quatre velocitats que 2 bits.

Cançons

La pestanya SO és la cançó oberta. Vuit columnes són els canals. Una cel·la és un identificador de patró. 255, dibuixat buit, significa que el canal està en silenci en aquesta fila. La columna X és una ordre: continuar, aturar, inici del bucle o final del bucle. Aturar finalitza la cançó després de la fila. Final del bucle salta enrere a l'últim inici del bucle.

Una fila acaba quan el patró de més a l'esquerra amb LOOP OUT a 0 acaba. Els patrons més a la dreta no allargaran aquesta fila: un de més llarg es talla, un de més curt torna a començar al seu primer pas fins que la fila acaba. Un patró amb LOOP OUT per sobre de 0 fa un bucle entre LOOP IN i LOOP OUT i no acaba la fila. Si tots els patrons de la fila fan un bucle, la fila es queda allà.

Dreceres

Clau	Acció

Tecles de piano	Escriu una nota sobre el pas seleccionat amb el ratolí.
Amunt / Avall	Canvia el valor que hi ha sota el cursor.
Espai	Reprodueix l'efecte sonor obert o la cançó oberta de la primera fila. Inicia/atura o reinicia des de la configuració de l'EDITOR D'EFECTES SONORS.
Ctrl+Z / Ctrl+Y	Desfer / refer.
Ctrl+C / Ctrl+X / Ctrl+V	Copia, talla i enganxa la selecció.

DADES

Taules amb nom. Cada taula està formada per files i columnes. Segons el codi, `dget`, `dset`, `dstr`, `drows` i `dcols` utilitzen aquestes taules. Els números de fila i columna del codi comencen a 0.

Tipus de columna, ciclats a la capçalera:

- **bool** — un bit.
- **2** — dos bits.
- **4** — quatre bits.
- **8** — un byte.
- **16** — 16 bits, per a identificadors i nombres més grans.
- **spr** — un identificador de sprite.
- **str** — text, dibuixat amb la font del sistema.
- **str8** i **str10** : el mateix text, dibuixat més alt.
- **f16** i **f32** : un flotador de mitja precisió i un flotador complet.

Amb el punter a la capçalera d'una columna, Ctrl+desplaçament canvia el tipus d'aquesta columna. Un desplaçament simple es mou per les columnes o cap avall per les files, segons el que seleccioni CONFIGURACIÓ → EDITOR DE DADES. Ctrl canvia aquesta direcció, per això Ctrl+desplaçament en una capçalera canvia el tipus quan la roda està configurada a columnes.

Les tecles de fletxa mouen. La tecla Intro edita una cel·la. CONFIGURACIÓ → EDITOR DE DADES:

- **Roda** — columnes de desplaçament o files de desplaçament.
- **Colors** : la quadrícula, la capçalera i la selecció.

Actius

ASSETS importa i exporta la peça que està oberta, no tot el projecte. Els sprites utilitzen l'sprite obert al PAINT, al full que està obert. El mapa és el mapa que està obert. SFX és el patró que està obert. La música és la cançó

que està oberta. Les dades són la taula que està oberta. El codi és l'excepció: s'escriu cada tabulació i es torna a llegir cada tabulació.

Amable	Fitxer	Què es mou
Sprites	`.png`	L'sprite obert.
Mapa	`.p37map`	El mapa obert.
Efectes sonors	`.p37sfx`	El patró obert.
Música	`.p37mus`	La cançó oberta. Els números de patrons es mantenen com estan. Exporteu els efectes sonors per separat si necessiteu aquests sons.
Dades	`.p37dat`	La taula oberta.
Codi	`.lua`	Cada pestanya.

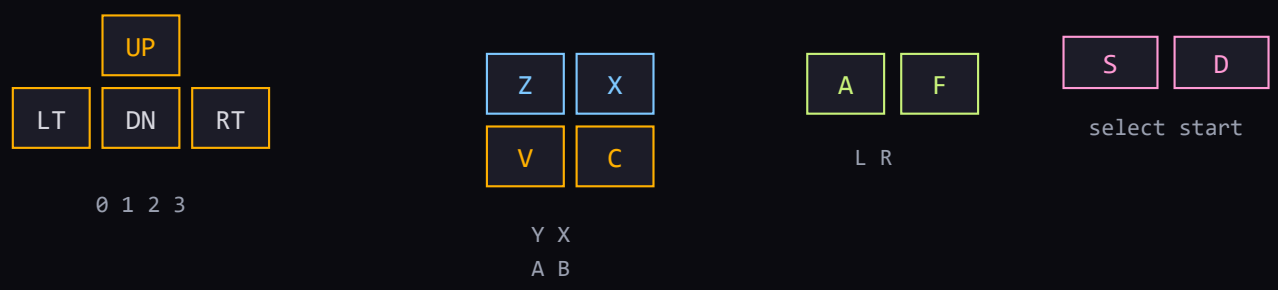
Configuració

La llista d'arrels:

Pàgina	Què emmagatzema
PERFILS	Desa i carrega tots els paràmetres. Crea'n de nous, canvia'n el nom, sobreescriu-los, suprimeix-los. Copia o enganxa el perfil com a text.
Menú	Finestra, tecles, colors de menú, fons opcional.
PANELL DE JOC	Vista prèvia i la finestra de reproducció.
EDITOR DE CODI	Destacat i pistes.
EDITOR DE PINTURA	Pinzell, paleta, vista de mapa.
EDITOR D'EFECTES SFX	Sintetitzador i entrada de notes.
EDITOR DE DADES	Direcció i colors de les rodes.
LLENGUA	Idioma de menús i pistes.
LLICÈNCIES	Textos de llicència, incloent-hi la font Fusion Pixel i SDL3.

GLOBAL:

- **DISPOSICIÓ** — disposició dels panells, a dalt.
- **VOLUM** — nivell mestre, de 0 a 10.
- **AJUSTAMENTS DE TECLES** : tecles per a cadascun dels 8 jugadors. Premeu una tecla per capturar-la. Indicacions, ABXYLR, seleccioneu.
- **Presentar** : ometre els fotogrames inactius o dibuixar tots els fotogrames.
- **Pantalla completa** — sense vores.
- **Actualitza** — FPS de la finestra. Pas esquerre i dret: 60, 90, 120 i màxim. Introdueix un número fins a 1000. 0 significa sense límit.
- **Vsync** — espera la visualització entre fotogrames.
- **Escala** — multiplicador de finestra entera.
- **Shaders** — CRT i filtres a l'editor, no al joc.
- **Color** : text del menú i ressaltat.
- **Fons** : un PNG opcional, retallat a 128, darrere del menú. Si s'activa "Esborra", s'elimina.
- **Cursor de l'editor** : forma del punter.
- **Ratolí** : punter flotant o enganxat a píxels.
- **Velocitat de la interfície d'usuari** : la rapidesa amb què es repeteix una direcció premuda a través d'una llista.
- **Explorador** : les llistes de fitxers comencen a l'arrel de l'ordinador o a la carpeta de l'exe.



Teclat per defecte. A és confirmar als menús. B és enrere. En un joc, aquests són els botons 4 i 5. Seleccionar (S) és el botó 10, no una drecera de menú.

Botó	Rol	Clau predeterminada
0	Esquerra	Fletxa esquerra
1	Dreta	Fletxa dreta
2	Amunt	Fletxa amunt
3	Avall	Fletxa avall
4	A, confirmar	V
5	B, esquena	C.

6	X	X
7	Y	Z
8	R	F
9	L	Un
10	Selecciona	S
11	Inici	D

Construir

Trieu una plataforma: Windows, Linux, Mac, Android, iOS o Web.

- **Nom** — nom del fitxer de la compilació.
- **Autor** — a Windows i Linux. Aquest és el nom de la carpeta on es desen els elements.
- **ID de l'aplicació Steam** : a Windows i Linux. Buit significa que no hi ha Steamworks. Un ID omplert assigna `achieve(0)`…`achieve(15)` als assoliments de Steam ACH_0…ACH_15.
- **Id** : identificador del paquet on la plataforma en demana un.
- **Amaga el codi** : a les capes d'escriptori i mòbils. Alenteix el `peek` del cartutx del joc. No és un bloqueig.
- **Pantalla completa** : el reproductor exportat comença a pantalla completa.
- **Shaders** : fins a tres capes integrades al reproductor.
- **Descarrega .p37** : només per a la web. Mostra una entrada de descàrrega al menú de pausa.
- **Icona** : una tessel·la d'un full de sprites: full, mida, x, y. O una portada PNG que retalles.
- **Exporta** — tria la carpeta. El menú indica COMPILACIÓ DESADA o COMPILACIÓ ERRORADA.

Fes una compilació amb Internet activat. El splash pot preguntar al servidor, de manera anònima, si es coneix el hash d'aquesta compilació. Aquesta comprovació hi és per dues raons. Em protegeix a mi, l'autor de PIX37, de còpies de l'eina que no eren meves per regalar-les. També protegeix la persona que va fer el joc. Agafar el joc d'una altra persona i llistar-lo com a propi s'està convertint en habitual al mercat de la telefonia, com ara Play (Android) o iOS. El joc encara s'inicia. Si la màquina està fora de línia, la introducció diu que el joc no està connectat. Aquesta línia també és un indici per al jugador que la còpia pot no haver vingut d'una font legal. És una petita comprovació, de manera que si el teu joc va ser robat i venut sense el teu consentiment per algú altre, pots fer aquest cas en una botiga com ara Play (Android), Steam o itch.io. El servidor emmagatzema l'hora en què es va registrar el hash per primera vegada, al costat del hash, i no el canvia més tard. L'hora anterior és la compilació original. Un hash registrat després d'això és una còpia posterior.

Vull que aquesta comprovació protegeixi les persones que creen jocs. El motor ja fa molta de la feina per ells, i aquesta és una d'aquestes coses. Entenc que pot ser controvertit. Si la comunitat decideix que és millor deixar aquest micro-DRM, ho faré.

Un cartutx de joc antic d'un editor anterior pot rebutjar l'arrencada després dels canvis d'idioma. Obriu el projecte `.pix37` en aquest editor i exporteu un nou `.p37`.

Connecta't a PIX37.PARTY

CONNECTA'T A PIX37.PARTY és dins de l'aplicació. No hi ha cap pàgina d'inici de sessió al navegador. Inicies la sessió amb un tiquet de sessió de Steam o amb itch.io, utilitzant el tauler i el ratolí. El servei mostra un nom per mostrar, de 2 a 16 caràcters, únic. La clau de llicència i l'ID de Steam no es mostren als altres jugadors.

Després dels termes del servei, el centre neuràlgic són Favorits, Catàleg, Classificació, Jugadors i el vostre perfil.

Una llicència de desenvolupador és l'editor. Tu fas el joc, el penges i hi deixes estrelles. Una llicència de jugador és el jugador. Els cors són de les persones que juguen en una ruta.

- **Catàleg** : navega per jocs. Els filtres cobreixen el gènere (fins a tres), la mida de la pantalla, la mida del cartutx del joc i si la càrrega inclou un projecte `.pix37`.
- **Preferits** : jocs que has marcat.
- **Pujada** : necessita una llicència de desenvolupador. Envieu un `.p37` i podeu adjuntar el projecte `.pix37`. Un `.p37` és un bytecode comprimit. És difícil de llegir i es pot ofuscar a propòsit. Un `.pix37` és el fitxer de projecte amb què realment treballeu: el codi font, les imatges i el so, en un format que una altra persona pot obrir i aprendre. Enviar-lo permet que altres creadors estudiïn el joc en lloc d'un cartutx de joc comprimit. El títol, l'autor, els gèneres, la portada, la icona i els noms dels assoliments l'acompanyen. El nombre de RAM enviat és el màxim que ha arribat al cartutx de joc.
- **Comentaris i assoliments** : a la pàgina del joc. El desbloqueig d'un assoliment mentre esteu connectats s'emmagatzema tant al compte com a la màquina.
- **Classificació** : desenvolupadors, ordenats per estrelles o per cors. Cada fila mostra els dos recomptes. Obriu un nom per veure els jocs d'aquest autor, ordenats per cors.
- **Jugadors** : comptes ordenats per assoliments desbloquejats.

Codi

El llenguatge és un petit script d'estil Lua. Aquesta meitat comença des d'un projecte buit, ensenya el llenguatge i després mostra un breu exemple de cada funció.

Un primer programa

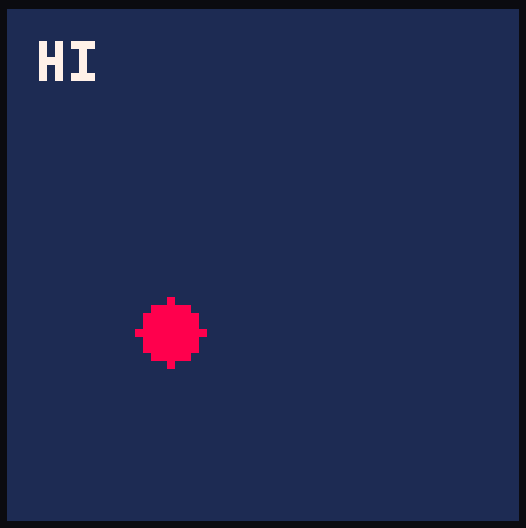
Crea un projecte a 64×64, color de 4 bits. Obre el CODI. Substitueix la pestanya per:

```
_init()
  x = 20
  y = 40
end

_update()
  if btn(0) then x -= 1 end
  if btn(1) then x += 1 end
  if btn(2) then y -= 1 end
  if btn(3) then y += 1 end
end

_draw()
  cls(1)
  print("HI", 4, 4, 7)
  circfill(x, y, 4, 8)
end
```

Ctrl+R l'executa. Les tecles de fletxa mouen el cercle. La tecla Esc atura el joc i torna a l'editor.



Les mateixes tres crides: `cls(1)` pinta la pantalla blava, `print` dibuixa HI i `circfill` dibuixa el cercle rosa.

Què funciona i quan

- `_init` s'executa una vegada.
- `_update` funciona a 60 Hz. Poseu el moviment i els botons aquí.
- `_draw` pinta. Sense cap argument fps, pinta tan ràpid com poden anar l'ordinador i el monitor. `_draw(30)` limita la pintura a 30 fps i deixa la lògica a 60.
- `_set_fps(n)` canvia aquest límit més tard. 0 l'esborra.

Els noms no distingeixen entre majúscules i minúscules; l'editor els emmagatzema en minúscules. `function` i `fun` són la mateixa paraula clau.

Llengua

Conceptes bàsics de programació

Un programa és una llista de línies que executa el cartutx del joc. Cada encapçalament mostra primer la forma de la línia i després un petit exemple. Una línia que comença amb `--` és un comentari: una nota per a tu. El joc la salta.

Variables

Una variable és un nom que conté un valor. Tu tries el nom. Les línies posteriors el poden llegir i poden emmagatzemar un nou valor amb el mateix nom.

La forma és `name = value`. `local name = value` manté aquest nom dins del bloc actual, de manera que la resta del cartutx del joc no el pot veure. Sense `local`, el nom és global i totes les pestanyes el poden veure.

```
hp = 3            -- global: the whole cart can read and change hp
local coins = 0   -- local: only this block can see coins
hp = hp + 1       -- read 3, add 1, store 4 back into hp
```

`nil` significa que no hi ha cap valor. `true` i `false` són sí i no. Un nom que no heu definit mai es llegeix com a `nil`.

```
bonus = nil            -- nothing stored yet
if bonus == nil then   -- condition: is bonus empty?
  bonus = 0            -- body: store a number
end
```

Taules

Una taula conté molts valors sota un sol nom. Una llista utilitza números. El primer element és 1, no 0. Un registre utilitza noms. Una taula pot contenir una altra taula.

Forma de la llista: `{ value, value }`. Forma del registre: `{ name = value, name = value }`. Una llista de registres és una taula els elements de la qual són taules.

```
-- a list. [1] is the first item
scores = {10, 20, 30}
print(scores[1], 1, 1, 7)      -- print draws text. this draws 10

-- a record. player.x is the field named x
player = {x = 10, y = 20}
print(player.x, 1, 8, 7)       -- draws 10

-- a list of records. enemies[1] is the first record
enemies = { {x = 4, y = 8}, {x = 20, y = 12} }
print(enemies[1].x, 1, 16, 7)  -- draws 4
print(enemies[2].y, 1, 24, 7)  -- draws 12
```

Condició i cos

Una condició és una pregunta amb una resposta afirmativa o negativa, com ara `hp > 0` o `btnd(4)`. Un cos és el grup de línies que s'executen quan s'entra en un bloc. `if` i tots els bucles utilitzen la mateixa forma: una condició, després un cos i després `end`. `repeat` es tanca amb `until` en lloc de `end`.

Si

`if` executa el seu cos només quan la condició és certa. `elseif` intenta una altra condició. `else` s'executa quan totes les condicions eren falses.

Forma: `if condition then`, després el cos i després `end`.

```
hp = 2
if hp > 0 then                 -- condition: is hp above zero?
  print("ok", 1, 1, 7)         -- body: print draws the word ok
elseif hp == 0 then            -- another condition
  print("empty", 1, 1, 7)
else
  print("no", 1, 1, 7)         -- body when every condition was false
end

if hp > 0 print(hp, 1, 32, 7)  -- one call, no then/end. draws 2
```

Bucles

Un bucle repeteix un cos. `while` comprova primer la condició i continua mentre es manté certa. `repeat` executa el cos i després ho comprova, de manera que el cos sempre s'executi almenys una vegada. `for` compta. S'inclou l'últim número. Sense un tercer número, el pas és 1, de manera que si l'inici ja ha passat el final, el cos no s'executa. El compte enrere necessita un pas negatiu.

```
n = 0
while n < 3 do         -- condition: keep going while n is under 3
  n += 1
  print(n)             -- 1, then 2, then 3: how many times the body has run
end

n = 0
repeat
  n += 1
  print(n)             -- body first, then the condition. prints 1, then 2, then 3
until n > 2

for i = 1, 3 do        -- i is 1, then 2, then 3
  print(i)
end

for i = 0, 4, 2 do     -- start, end, step of 2
  print(i)             -- 0, then 2, then 4
end

for i = 3, 1, -1 do    -- counting down needs the step
  print(i)             -- 3, then 2, then 1
end

for i = 1, 5 do
  if i == 3 then break end  -- leave the loop when i reaches 3
  print(i)                  -- 1, then 2
end
```

Caminant per una taula

`all` visita cada element d'una llista, des del primer. `pairs` visita cada nom i el valor emmagatzemat sota aquest nom.

```
t = {1, 2, 3}                -- list: t[1] is 1, t[2] is 2, t[3] is 3
for v in all(t) do           -- v is 1, then 2, then 3
  print(v)                   -- print draws that number
end

player = {x = 10, y = 20}    -- record: a field x and a field y
for k, v in pairs(player) do
  print(k)                   -- k is the name, "x" then "y"
end
```

Funcions

Una funció és una recepta amb nom. La forma és `function name(arguments)`, després un cos i, finalment, `end`. La crideu escrivint el nom i els parèntesis. `return` envia un valor de retorn i pot enviar-ne més d'un. Els arguments són locals d'aquesta funció.

```
function greet(who)       -- who is a local that holds what you pass in
  return "hi "..who       -- .. glues two texts. return sends the result back
end

print(greet("ada"))       -- draws hi ada
```

```
function pair()
  return 1, 2             -- two values
end

a, b = pair()             -- a becomes 1, b becomes 2
```

`local function` és una funció que pot veure els elements locals que l'envolten. Això és un tancament. Una taula en pot contenir un, i s'invoca amb un punt.

```
function make()
  local n = 0
  local m = {}
  m.bump = function()     -- this function sees n
    n += 1
    return n
  end
  return m
end

counter = make()
print(counter.bump())     -- 1
print(counter.bump())     -- 2
```

Sí, no, i triar

`not` inverteix una condició. `and` només és cert quan ambdues cares són certes. `or` és cert quan qualsevol de les cares és certa. `a and b or c` tria `b` quan `a` és cert, en cas contrari `c`. `cond ? a : b` és la mateixa idea en una línia: `a` quan la condició és certa, `b` quan no ho és.

```
hp = 2
hit = false
night = false

if hp > 0 and not hit then         -- both parts are true, so the body runs
  print("safe", 1, 1, 7)           -- draws safe
end

label = hp > 0 and "ok" or "out"   -- hp is above 0, so label is ok
print(label, 1, 8, 7)

shade = night ? 1 : 7              -- night is false, so shade is 7
print(shade, 1, 16, 7)
```

Matemàtiques i text

`+ - * /` són sumes, restes, multiplicacions i divisions. `//` divideix i conserva el nombre enter. `%` és el residu. `^` és una potència. `..` uneix dos textos. `#t` és el nombre d'elements que hi ha en una llista. Les comparacions són `==` (igual), `~=` o `!=` (diferents) i `< > <= >=`. `x += 1` suma 1 a x. La mateixa forma funciona per a `-= *= /= ^= %=`.

```
scores = {10, 20, 30}    -- the list named scores
tiles = 15 // 8          -- 1, the fraction is dropped
left = 15 % 8            -- 7, what is left over
title = "hi "..."ada"    -- the text hi ada
n = #scores              -- how many items are in the list named scores
print(n, 1, 1, 7)        -- 3
print(title, 1, 8, 7)

score = 0
best = 10
if score ~= best then    -- condition: score is not the best
  score += 10            -- add 10 to score
end
print(score, 1, 16, 7)   -- 10
```

On viu un local

`local` dura fins al `end` que tanca el seu bloc. `if` / `end` és el seu propi bloc. Podeu esmentar un local a sobre de la línia que el defineix, però fins que aquesta línia no s'executa el valor és `nil`.

```
if true then
  print(coins)       -- nil: the next line has not run yet
  local coins = 5
  print(coins)       -- 5
end
-- coins does not exist out here
```

Saltant

`::label::` marca una línia. `goto label` salta allà, però només dins de la mateixa funció.

```
x = 0
::again::
x += 1                      -- add 1
if x < 3 then goto again end  -- condition: jump back while x is under 3
```

La memòria, en resum

`peek` i `poke` miren els píxels actius a través d'una adreça de 16 bits. No conserven una segona còpia del cartutx del joc. Les adreces `0x5F00` a `0x5FFF` són registres: càmera, clip (només lectura), color de dibuix, patró d'ompliment, les tres remapatges de la paleta i els números de pàgina. Els bytes a banda i banda d'aquest forat, 65280 en total, són la finestra de píxels.

Quan la pantalla cap en aquests 65280 bytes, 64×64 i 128×128 sí i 256×256 no, la pantalla comença a l'adreça 0, fila per fila. El que queda és el full de sprites seleccionat per `0x5F41` (0 és el full 0). Si aquest full és més gran que l'espai restant, `0x5F42` selecciona la pàgina i el mapa no és a la finestra. Si el full cap, la cua són les cel·les del mapa seleccionat per `0x5F43`, un o dos bytes cadascuna, i `0x5F44` pagina el mapa quan no cap a la cua.

Quan la pantalla no cap, tota la finestra és una pàgina de 65280 píxels. `poke(0x5F40, n)` gira la pàgina de la pantalla. `poke(0x5F41, n)` amb n diferent de 0 mostra el full n en comptes d'això, i el full 0 no té vista en aquest mode. `poke(0x5F43, n)` amb n diferent de 0 mostra el mapa n, i això guanya el full. El mapa 0 en aquest mode només és `mget` i `mset`.

Un `peek` que no aterra en un píxel o un registre retorna 0. Un `poke` que no hi fa res. Els indicadors de sprite, SFX i taules DATA no són en aquesta finestra. Utilitzeu `fget`, `sfxget` i `dget`.

Ús pràctic

Totes les funcions apareixen a la part inferior d'aquesta pàgina, amb usos pràctics de les funcions del PIX37. Els enllaços que hi ha sota cada exemple donen més informació sobre les funcions.



Moviment suau

Un cartutx de joc de 128×128, vist des de dalt. L'sprite 1 ja està dibuixat. Les fletxes acceleren en qualsevol direcció i la fricció et frena. L'sprite s'enrotlla: una meitat que surt de la part inferior es dibuixa a la part superior, i el mateix passa amb les altres vores.

`_init` `_update` `_draw` `btn` `mid` `cls` `spr` `palt`



Comprovació de si el següent moviment està bloquejat

Un cartutx de joc de 64×64. El terra i la paret dreta ja són al mapa, pintats amb l'sprite 3, i la bandera 0 està establerta en aquest sprite. Les fletxes mouen l'sprite 1. La paraula a la part superior diu bloquejat quan el següent pas se superposaria a una rajola sòlida i lliure quan no ho faria.

`flr` `fget` `mget` `btn` `cls` `map` `spr` `print` `_update` `_draw`



Una neteja amb gir

Premeu A. El wipe es dibuixa amb flip(), una fila negra a la vegada, en comptes de dins de _draw(). _draw es manté fora del camí mentre s'executa, de manera que cada fila roman a la pantalla.

`rectfill` `flip` `btnd` `cls` `print` `_update` `_draw`



Una taula de funcions (Classe)

L'Sprite 1 ja està dibuixat. make_monster funciona com una classe. Retorna una taula, i update i draw són mètodes que mantenen la x i la y pròpies del monstre. El número de la part superior és la distància que ha recorregut.

`spr` `cls` `print` `flr` `_init` `_update` `_draw`



Quatre maneres de llegir un botó

A és el botó 4. Mantingueu-lo premut i la càrrega puja. Deixeu-lo anar i la càrrega torna a 0. Amunt i avall es repeteixen mentre es manté premut, de manera que la fila del menú continua movent-se. btnd és el primer fotograma d'una pulsació, que aquest cartutx de joc només utilitza com a inici d'aquesta pulsació.

`btn` `btnd` `btnp` `btnu` `cls` `print` `_update` `_draw`



Dispara amb el ratolí

L'Sprite 1 ja està dibuixat al mig. Un punt blanc és el cursor. Si feu clic amb el botó esquerre, es dispara un píxel cap a ell. La roda canvia la velocitat, que s'imprimeix a la cantonada.

`scroll` `clickl` `mouse` `sqrt` `add` `all` `cls` `spr` `pset` `print` `_update` `_draw`



Escriu un nom i confirma'l

Premeu A per obrir el camp. Les tecles amunt i avall canvien una lletra que parpelleja. A l'escriu, X l'elimina, Y és un espai i B la confirma. Aleshores, la línia diu el vostre nom.

`btnd` `keyon` `keystr` `flr` `time` `cls` `print` `_update` `_draw`



Càmera al reproductor

Un cartutx de joc de 128×128. L'sprite 1 comença al mig d'un terra de rajoles. Les fletxes es mouen. La càmera manté l'sprite centrat. x i y a la cantonada no es desplacen, perquè camera() sense arguments torna a posar la vista per al HUD.

`btn` `screen` `camera` `cls` `map` `spr` `print` `flr` `_update` `_draw`



Unes poques línies d'una novel·la visual

Les línies són en DATA, en anglès i polonès. A continua o tria l'opció "queda" quan hi ha una opció a la pantalla. B tria l'opció "surt" i, en cas contrari, canvia l'idioma. X canvia l'idioma mentre es mostra una opció. El diàleg utilitza la font de 8px perquè les lletres poloneses es dibuixin.

`dstr` `dget` `btnd` `cls` `print8` `print` `_update` `_draw`



Pinta amb el picar i el tacar

Un cartutx de joc de 64×64, de manera que l'adreça 0 és el píxel superior esquerre i la següent fila comença a 64. El cartutx de joc escriu píxels, copia una fila, esborra una fila i imprimeix els valors que ha llegit.

`cls` `poke` `peek` `poke2` `peek2` `poke4` `peek4` `memcpy` `memset` `print` `_draw`



Una llista a la qual pots afegir

A suma un número, fins a 8. B elimina l'últim. X elimina el primer. La llista s'imprimeix a la pantalla.

`btnd` `count` `add` `deli` `cls` `ipairs` `print` `pack` `unpack` `next` `foreach` `select` `pcall` `assert` `del` `_update` `_draw`



Un nom desmuntat

El cartutx del joc divideix ada, 12, blue i imprimeix el nom, la seva primera lletra, 12, i la paraula blue.

`cls` `split` `sub` `chr` `ord` `tonum` `tostr` `print` `_draw`



Toca una nota i després canvia-la

El patró 0 té una nota de llargada i cada repetició té la mateixa intensitat. A el torna a reproduir quan aquesta nota ha acabat. B puja el to i reprodueix aquesta nota.

`sfxset` `sfxhdr` `sfx` `btnd` `sfxstat` `sfxget` `cls` `print` `_init` `_update` `_draw`

 Una cançó escrita al cartutx del joc

Dos patrons es repeteixen en bucle a la fila 0, un a cadascun dels dos primers canals. La fila es manté a 0. ch0 i ch1 són la nota en què es troba cada canal. Editar el patró, LEN i LOOP al cartutx del joc és una manera de veure què fan aquests controls al panell de música.

`songget` `songset` `music` `musicstat` `cls` `print` `_init` `_draw`

 Tres monstres a DATA

El panell DADES ja té una columna de nom i una columna de punts de vida, amb el slime, el ratpenat i l'ull. A fa mal al slime. Els números de la dreta són els seus punts de vida.

`drows` `dstr` `dget` `add` `btnd` `dset` `dcols` `cls` `print` `all` `_init` `_update` `_draw`

 Recorda la puntuació

A afegeix un punt i el guarda. Surt i torna a obrir el cartutx del joc: el número encara hi és. B l'oblida.

`load` `btnd` `save` `delsave` `cls` `print` `_init` `_update` `_draw`

  Un segon capítol en un altre fitxer

Dos cartutxos de joc comparteixen la missió de la ranura de desament. Part 1: A marca el joc acabat. La part 2, oberta primer, diu que s'ha d'acabar la part 1. Després d'això, s'obre la part 2 i B esborra la partida desada perquè es pugui tornar a jugar a la part 1.

`load` `saveslot` `btnd` `save` `delsave` `cls` `print` `_init` `_update` `_draw`

 Un assoliment

L'sprite 8 ja està dibuixat. Premeu A deu vegades. El toast anomena l'assoliment i l'sprite 8 es queda perquè està desbloquejat. B esborra aquest assoliment del fitxer local i reinicia el recompte.

`achdef` `btnd` `achieved` `achieve` `achclear` `cls` `print` `spr` `_init` `_update` `_draw`

 Una habitació on puguis entrar

Un cartutx de joc de 256×256. Un punt blanc és el cursor. El menú és host, join o la llista de servidors. Host obre una sala anomenada walk i inicia el joc. Join desplaça una adreça (dígits, un punt, dos punts) i B es

connecta. La llista de servidors es mou amunt i avall, i A s'uneix a la fila. Al joc, les fletxes caminen per l'sprite 1. Es comparteix una moneda: l'host la mou i, en recollir-la, s'afegeix 1 a la puntuació d'aquest jugador i es genera el següent. Cada sprite, inclòs el teu, mostra aquesta puntuació al costat. La cantonada mostra quants jugadors hi ha, l'adreça LAN i l'adreça pública de l'host. B marxa: tu i els altres torneu al menú i el següent host comença una nova sala amb la puntuació 0.

`btn` `btnd` `mouse` `pset` `spr` `print` `flr` `rnd` `pairs` `nlisten` `nhost` `nannounce` `nlist` `njoin` `nleave` `nsend` `nkb` `nkbstr` `whoami` `nplayers` `nlan` `_mp` `_mp_join` `_mp_leave` `_mp_fail` `palt` `cls` `time` `_init` `_update` `_draw`



Diàleg que espera

A mostra la línia següent. La funció s'atura a cada "ceed" fins que torneu a prémer A.

`yield` `cocreate` `btnd` `coresume` `cls` `print` `_init` `_update` `_draw`



Espera mig segon.

La paraula canvia de ... a hi després d'aproximadament mig segon. La pantalla continua dibuixant mentre la corrutina està en pausa.

`yield` `cocreate` `costatus` `coresume` `cls` `print` `_init` `_update` `_draw`



Cercles que arriben un per un

Cinc cercles degoten des de dalt, a dos fotogrames de distància, i després cauen.

`add` `yield` `cocreate` `coresume` `all` `cls` `circfill` `_init` `_update` `_draw`



Dos fotogrames, del rellotge

L'sprite 1 i l'sprite 2 són dos fotogrames del mateix personatge. La imatge s'intercanvia quatre vegades per segon.

`cls` `flr` `time` `spr` `_draw`

Cada funció

Els arguments entre claudàtors són opcionals. Els índexs de taules i cadenes comencen a 1, excepte les files i columnes DATA, que comencen a 0. Els colors són índexs de paleta.

Índex

`_dibuixar` `_mp` `_mp_join`
`_inici` `_mp_fail` `_mp_abandonar`

_mp_actualització
_set_fps
_actualitzar
abs
achclar
achdef
aconseguir
aconseguit
afegir
tots
afirmar
atan2
banda
bnot
bor
btn
btnd
btnp
btnu
bxor
càmera
sostre
chr
circ
circfill
feu clic
clicm
clickr
clip
cls
cocrear
color
coresume
cos
costatus
comptar
cursor
dcols
del
delicatessen
delsave
dget
drows
dset
dstr
fget
omplir
voltejar
flr
foreach
fset
informació
ipairs
clau apagada
clau
keystat
keystr
línia
càrrega
registre
lshr
mapa
màx
memcpy
memset
mget
mig
min
ratolí
mset
música
musicstat
nananunciar
ncall
següent
nhost
njoin
nkb
nkbstr
nlan
n'abandonar
nlista
nescolta
nota
njugadors
nsend
nstun
ord
ovalada
ovalfill
paquet
parelles
amic
palrgb
palt
pcall
ullada
ullada 2
ullada4
perf
pàg
picar
picar2
picar4
imprimir
imprimir 10
imprimir8
pset
rect
rectificar
rnd
rotl
rotr
correcte
omplir correctament
guardar
ranura de desat
pantalla
desplaçar-se
seleccionar
setmap
setpalette
setspr

setex
sectorn
sfx
sfxget
sfxhdr
sfxset
sfxstat
sget
sgn
shl
estadístiques
shr
pecat
cançó
conjunt de cançons
dividir
spr
quadrada
srand
sset
sspr
estat
sub
temps
tline
tonum
tostr
desempaquetar
whoami
rendiment

Cicle de vida

_init

`_init() ... end`

S'executa una vegada quan s'inicia el cartutx del joc. El codi que es troba fora de qualsevol hook també es plega a _init.

```
_init()
  x = 64
  y = 40
end
```

_update

`_update() ... end`

Lògica del joc. Es crida a 60 Hz, fins i tot quan _draw està limitat o l'amfitrió és més lent. Llegiu els botons aquí.

```
_update()
  if btn(0) then x -= 1 end
  if btn(1) then x += 1 end
end
```

_draw

`_draw([fps]) ... end`

Dibuixa un fotograma. Sense cap argument, o amb 0, s'executa tan sovint com es presenta l'amfitrió. Passa n (1..999) per limitar el dibuix a n fotogrames per segon. _update es manté a 60 Hz.

```
_draw(30)
  cls(1)
  spr(1, x, y)
end
```

_set_fps

`_set_fps(n)`

Canvia el límit de _draw mentre el joc està en execució. 0 significa sense límit. n està fixat a 0..999.

```
if btn(4) then _set_fps(60) end
```

flip

`flip()`

Presenta la pantalla i cedeix fins al següent fotograma. Només s'utilitza si es dibuixa el propi bucle de fotogrames. No s'espera després.

```
cls(0)
print("tick", 4, 4, 7)
flip()
```

time

`time()`

Segons des de _init.

```
print(time(), 4, 4, 7)
```

Entrada

btn

`btn([b], [p])`

1 mentre es manté premut el botó b, altrament 0. p és el jugador, 0..7. Sense arguments retorna una màscara de bits del jugador 0. Botons: 0 esquerra, 1 dreta, 2 amunt, 3 avall, 4 A, 5 B, 6 X, 7 Y, 8 R, 9 L, 10 seleccionar, 11 iniciar.

```
if btn(4) then shoot() end
```

btnd

`btnd(b, [p])`

Cert només al marc _update quan es prem el botó b.

```
if btnd(5) then menu_open = not menu_open end
```

btnp

`btnp([b], [p])`

Cert en prémer, i de nou mentre es manté premut: 30 tics de retard, i després cada 8 tics. Sense botó, retorna una màscara.

```
if btnp(2) then cursor -= 1 end
```

btnu

`btnu(b, [p])`

Cert només al marc on es deixa anar el botó.

```
if btnu(4) then charge = 0 end
```

mouse

```
mouse()
```

Retorna els valors x i y del ratolí en els píxels del joc.

```
mx, my = mouse()
pset(mx, my, 7)
```

clickl

```
clickl([p])
```

Botó esquerre del ratolí. Es manté premut clickl(). clickl(1) és la vora de premsa.

```
if clickl(1) then place(mouse()) end
```

clickr

```
clickr([p])
```

Botó dret del ratolí. Mateixa vora d'1 argument que amb clicl.

```
if clickr() then cancel() end
```

clickm

```
clickm([p])
```

Botó central o un clic de la roda de desplaçament.

```
if clickm(1) then drag = true end
```

scroll

`scroll()`

Moviment de la roda del ratolí en aquest fotograma. Positiu i negatiu depenen de la direcció de la roda.

```
cam_y -= scroll()
```

keyon

`keyon()`

Obre un camp de text controlat per botons. Amunt i avall canvien una lletra de la A a la Z. A escriu aquesta lletra. X elimina. Y és un espai. B confirma i tanca el camp. El teclat no hi escriu.

```
if btnd(4) then keyon() end
```

keyoff

`keyoff()`

Amaga el camp de text.

```
keyoff()
```

keystr

`keystr([s])`

Text al camp, després 1 al marc B confirma, i després la lletra amunt i avall canvia. Aquesta lletra queda buida un cop es tanca el camp. keystr("") esborra el text.

```
name, done, letter = keystr()
if done == 1 then start(name) end
```

keystat

`keystat()`

1 mentre el camp de text està obert, altrament 0.

```
if keystat() == 1 then return end
```

Dibuix

screen

`screen()`

Mida de pantalla del cartutx de joc actual en píxels. Retorna l'amplada i després l'alçada. Aquesta és la resolució del joc, no la finestra de l'editor.

```
w, h = screen()
```

cls

`cls([col])`

Esborra la pantalla. El color per defecte és 0.

```
cls(1)
```

color

`color(col)`

Estableix el color de dibuix utilitzat quan una crida posterior omet col. Sense cap argument, retorna el color actual.

```
color(7)
print("hi")
```

camera

`camera([x, y])`

Desplaça cada crida de dibuix. camera() es reinicia a 0, 0.

```
camera(cam_x, cam_y)
```

clip

`clip([x, y, w, h, clip])`

Limita el dibuix a un rectangle. Un cinquè argument diferent de zero interseca amb el clip actual. clip() restaura la pantalla completa. La vista prèvia dels registres del clip és de només lectura.

```
clip(8, 8, 48, 48)
circfill(32, 32, 20, 8)
clip()
```

cursor

`cursor([x, y, col])`

Estableix on va la següent impressió sense coordenades i, opcionalment, el color. cursor() retorna x, y.

```
cursor(4, 4, 7)
print("line 1")
print("line 2")
```

pset

`pset(x, y, [col])`

Defineix un píxel de pantalla. Compta per a l'estadística de px.

```
pset(x, y, 10)
```

pget

`pget(x, y)`

Color del píxel de la pantalla a x, y.

```
if pget(x, y+8) == 3 then grounded = true end
```

print

`print(v, [x, y, col])`

Dibuixa text de 3×5 (UTF-8, retallat de la font Fusion). Sense x ni y, s'imprimeix al cursor i es mou a la línia següent.

```
print("score "..score, 2, 2, 7)
```

log

`log(...)`

Afegeix els arguments, separats per espais, a una memòria intermèdia de text. pix37_run imprimeix aquesta memòria intermèdia a la sortida estàndard. No dibuixa.

```
log("hp", hp)
```

print8

`print8(v, [x, y, col])`

Igual que la impressió, amb la font Fusion de 8 px.

```
print8("pause", 16, 28, 7)
```

print10

`print10(v, [x, y, col])`

Igual que la impressió, amb la font Fusion de 10 px.

```
print10("pix37", 8, 8, 10)
```

line

`line([x0, y0, x1, y1, col])`

Línia de x0,y0 a x1,y1. Dos arguments continuen des del final anterior. line() reinicia aquesta cadena.

```
line(0, 32, 63, 32, 6)
```

circ

`circ(x, y, r, [col])`

Contorn del cercle. El radi és en píxels.

```
circ(32, 32, 12, 12)
```

circfill

`circfill(x, y, r, [col])`

Cercle omplert.

```
circfill(x, y, 3, 8)
```

oval

`oval(x0, y0, x1, y1, [col])`

Contorn de l'el·lipse dins del quadre delimitador. Les cantonades són inclusives.

```
oval(8, 20, 56, 44, 7)
```

ovalfill

`ovalfill(x0, y0, x1, y1, [col])`

El·lipse plena en un quadre delimitador.

```
ovalfill(10, 40, 54, 60, 3)
```

rect

`rect(x0, y0, x1, y1, [col])`

Contorn rectangular. Les cantonades són inclusives.

```
rect(0, 0, 63, 63, 7)
```

rectfill

`rectfill(x0, y0, x1, y1, [col])`

Rectangle omplert.

```
rectfill(0, 56, 63, 63, 3)
```

rrect

`rrect(x, y, w, h, [r, col])`

Contorn de rectangle arrodonit. x i y són la part superior esquerra. w i h són la mida en píxels; tots dos han de ser superiors a 0. r és el radi de la cantonada i per defecte és 0. Està fixat a la meitat del costat més curt.

```
rrect(8, 8, 48, 16, 3, 7)
```

rrectfill

`rrectfill(x, y, w, h, [r, col])`

Rectangle arrodonit i omplert. Mateixos arguments que rrect.

```
rrectfill(8, 40, 48, 16, 2, 1)
```

fillp

`fillp([p])`

Patró d'ompliment de 16 bits per a formes omplides posteriorment. 0 és sòlid. fillp() retorna el patró actual. Els bits recorren el patró de 4×4.

```
fillp(0xA5A5)
rectfill(8, 8, 40, 40, 10)
fillp(0)
```

pal

`pal([c0, c1, [mode]])`

Reassigna el color c0 a c1. El mode 0 és la paleta de dibuix, 1 la paleta de visualització, 2 la paleta de patró d'ompliment. pal() es reinicia.

```
pal(8, 10)
circfill(32, 32, 8)
```

palrgb

`palrgb(col, r, g, b)`

Estableix una entrada de la paleta a RGB, cada canal de 0 a 255.

```
palrgb(8, 255, 80, 40)
```

palt

`palt([col, t])`

Transparència. palt(col) o palt(col, true) fa que el col sigui transparent. palt(col, false) el fa sòlid. palt() fa que tots els colors siguin opacs.

```
palt(0)
spr(1, x, y)
```

setpalette

`setpalette(n)`

Selecciona el banc de paletes n. setpalette() retorna el banc actual.

```
setpalette(1)
```

tline

`tline(x0, y0, x1, y1, mx, my, [mdx, mdy])`

Dibuixa una línia a la pantalla mentre mostra els píxels del mapa des de la posició del mapa mx, my, mdx i mdy, i desplaça el exemple del mapa al llarg de la línia.

```
tline(0, 40, 63, 20, 0, 0, 0.5, 0)
```

Sprites i mapes

spr

`spr(n, x, y, [w, h, fx, fy, tex, rot])`

Dibuixa l'sprite n a x, y. w i h són la mida en píxels, per defecte 8. fx i fy s'inverteixen horitzontalment i verticalment quan són diferents de zero. tex és la tabulació del full d'sprite. rot és l'angle en girs (0..1), mateixa direcció que sin i cos. 0 deixa l'sprite sense girar.

```
spr(1, x, y, 8, 8, 0, 0, 0, aim)
```

sspr

`sspr(sx, sy, sw, sh, dx, dy, [dw, dh, fx, fy, tex])`

Copia un rectangle del full de sprites a la pantalla, escalat a dw per dh. Els valors per defecte mantenen la mida original.

```
sspr(0, 0, 8, 8, 16, 16, 32, 32)
```

sget

`sget(x, y, [tex])`

Color d'un texel al full de sprites.

```
c = sget(0, 0)
```

sset

`sset(x, y, [col, tex])`

Escriu un texel al full de sprites.

```
sset(3, 3, 8)
```

fget

`fget(n, [f, tex])`

Indicadors de sprite. fget(n) retorna el byte de l'indicador. fget(n, f) és cert quan el bit d'indicador f (0..7) està establert.

```
if fget(tile, 0) then solid = true end
```

fset

`fset(n, [f], v, [tex])`

Estableix els indicadors d'sprite. fset(n, v) escriu el byte sencer. fset(n, f, v) estableix un bit.

```
fset(2, 0, true)
```

setspr

`setspr(n)`

Selecciona el full de sprites actiu. setspr() retorna el full actual. El mateix banc que settex.

```
setspr(1)
```

settex

`settex(n)`

Àlies de setspr.

```
settex(0)
```

map

`map(cx, cy, sx, sy, [cw, ch, layer, tex])`

Dibuixa el mapa. La cel·la cx, cy del mapa es dibuixa a la pantalla sx, sy. cw, ch de 0 significa que tot el mapa. La capa s'accepta i s'ignora. tex és el full del conjunt de tessel·les.

```
map(0, 0, 0, 0, 16, 16)
```

mget

`mget(cx, cy)`

ID de sprite a la cel·la del mapa cx, cy.

```
if fget(mget(tx, ty), 0) then bounce() end
```

mset

`mset(cx, cy, n)`

Escriu l'sprite n a la cel·la del mapa cx, cy.

```
mset(3, 4, 0)
```

setmap

`setmap(n)`

Selecciona el mapa n. setmap() retorna el mapa actual. El mapa 0 és un mapa real.

```
setmap(1)
```

Memòria

peek

`peek(addr)`

Llegeix un byte de la finestra de 64 KB. La finestra és la pantalla en directe, després els píxels del sprite i, a continuació, les cel·les es mapegen quan la pantalla cap en 65280 bytes. Les pantalles més grans es pagina amb poke(0x5F40, page). Les adreces que no estan mapejades retornen 0. També s'escriu @addr.

```
px = peek(0)
```

poke

`poke(addr, ...)`

Escriu bytes a addr. Els arguments addicionals escriuen els següents bytes. Les adreces no assignades s'ignoren. També s'escriu $addr, val. Picar la pantalla compta com una escriptura de píxel.

```
poke(0x5F00, 4) -- camera x
```

peek2

`peek2(addr)`

Valor de 16 bits sense signe, little-endian, a l'adreça.

```
w = peek2(0x6000)
```

poke2

`poke2(addr, val)`

Escriu un valor de 16 bits sense signe, little-endian.

```
poke2(0x6000, 1000)
```

peek4

`peek4(addr)`

Valor de 32 bits sense signe, little-endian.

```
n = peek4(0x6000)
```

poke4

`poke4(addr, val)`

Escriu un valor de 32 bits sense signe, little-endian.

```
poke4(0x6000, 1)
```

memcpy

`memcpy(dst, src, len)`

Copia len bytes dins de la mateixa finestra de 64 KB.

```
memcpy(0x6000, 0x6100, 64)
```

memset

`memset(dest, val, len)`

Omple els bytes de len començant per dest amb val.

```
memset(0, 0, 64)
```

Matemàtiques

abs

`abs(x)`

Valor absolut.

```
d = abs(tx - x)
```

flr

`flr(x)`

Arrodoneix cap a menys infinit.

```
tx = flr(x / 8)
```

ceil

`ceil(x)`

Arrodoneix cap a l'infinit.

```
n = ceil(hp / 8)
```

max

`max(a, b)`

El més gran dels dos valors.

```
x = max(0, x)
```

min

`min(a, b)`

El més petit dels dos valors.

```
x = min(63, x)
```

mid

`mid(a, b, c)`

El mig dels tres valors. Útil com a pinça.

```
x = mid(0, x, 63)
```

sgn

`sgn(x)`

Signe de x: -1, 0 o 1.

```
x += sgn(dir)
```

sqrt

`sqrt(x)`

Arrel quadrada. Un argument negatiu no és un nombre.

```
dist = sqrt(dx*dx + dy*dy)
```

sin

`sin(x)`

El sinus x és en torns: 0, 0,25, 0,5, 0,75 donen una volta al cercle. No en radians.

```
y = 32 + sin(time()) * 8
```

cos

`cos(x)`

El cosinus, al seu torn, té la mateixa convenció que el sin.

```
x = 32 + cos(time()) * 8
```

atan2

`atan2(dx, dy)`

Angle del vector dx, dy, en torns de 0 a 1.

```
aim = atan2(mx - x, my - y)
```

rnd

```
rnd([x])
```

Nombre aleatori en [0, x). Per defecte x és 1, per tant rnd() és 0..1. Per a un dau, flr(rnd(6))+1.

```
c = flr(rnd(16))
```

srand

```
srand(seed)
```

Sembra el generador aleatori perquè es pugui repetir una seqüència rnd posterior.

```
srand(1)
```

band

```
band(a, b)
```

I bit a bit. També s'escriu a i b.

```
if band(flags, 1) ~= 0 then hit() end
```

bor

```
bor(a, b)
```

O bit a bit. També s'escriu a | b.

```
flags = bor(flags, 4)
```

bxor

```
bxor(a, b)
```

XOR bit a bit. També s'escriu a ^^ b.

```
n = bxor(n, 255)
```

bnot

`bnot(a)`

NOT a nivell de bit. També s'escriu ~a.

```
mask = bnot(1)
```

shl

`shl(a, n)`

Majúscula a l'esquerra. També s'escriu una << n.

shr

`shr(a, n)`

Desplaçament aritmètic a la dreta. També s'escriu amb una >> n.

```
n = shr(n, 1)
```

lshr

`lshr(a, n)`

Desplaçament lògic a la dreta. També escrit com a >>> n.

```
n = lshr(n, 4)
```

rotl

`rotl(a, n)`

Gira els bits cap a l'esquerra. També s'escriu una <<> n.

```
n = rotl(n, 1)
```

rotr

`rotr(a, n)`

Gira els bits a la dreta. També s'escriu com a >>< n.

Taules i valors

add

`add(t, v, [i])`

Afegeix v a la taula t. Si es proporciona i, insereix en aquest índex basat en 1.

```
add(bullets, {x=x, y=y})
```

del

`del(t, v)`

Elimina la primera entrada igual a v.

```
del(items, gem)
```

deli

`deli(t, [i])`

Elimina l'índex i i retorna aquest valor. Sense i, elimina l'última entrada.

```
local b = deli(bullets, i)
```

all

`all(t)`

Iterador sobre la part de la matriu, en ordre. Els forats aturen el recorregut en la manera com acaba una matriu basada en 1.

```
for b in all(bullets) do
  b.x += 2
end
```

foreach

`foreach(t, fn)`

Crida a fn(v) per a cada element de la matriu.

```
foreach(enemies, draw_enemy)
```

pairs

`pairs(t)`

Iterador sobre cada clau i valor, inclosos els camps amb nom.

```
for k, v in pairs(player) do
  print(k)
end
```

ipairs

`ipairs(t)`

Iterador sobre la part de la matriu amb l'índex basat en 1.

```
for i, v in ipairs(deck) do
  spr(v, i*8, 8)
end
```

next

`next(t, [k])`

Retorna la clau i el valor següents després de k. next(t) comença al principi. S'utilitza per parells.

```
k, v = next(t)
```

count

`count(t, [v])`

Longitud de la part de la matriu. Si es dóna v, compta quantes entrades són iguals a v.

```
if count(gems) == 0 then win() end
```

pack

`pack(...)`

Empaqueta els arguments en una nova taula.

```
t = pack(1, 2, 3)
```

unpack

`unpack(t, [i, j])`

Retorna de t[i] a t[j] com a valors separats. Els valors per defecte cobreixen tota la part de la matriu.

```
a, b = unpack(t)
```

select

`select(i, ...)`

select("#", ...) retorna quants arguments segueixen. select(i, ...) retorna els arguments des de i en endavant.

```
n = select("#", 1, 2, 3)
```

assert

`assert(c, [msg])`

S'atura amb un error quan c és fals o nul. msg substitueix el text per defecte.

```
assert(hp > 0, "dead")
```

pcall

`pcall(f, ...)`

Crida f. En cas d'èxit retorna true i els resultats. En cas d'error retorna false i el text de l'error.

```
ok, err = pcall(load_level, 3)
```

Cordes

chr

`chr(n)`

Un caràcter d'un bytecode 0..255.

```
s = chr(65) -- A
```

ord

`ord(s, [i, j])`

Codis de bytes de s des de l'índex i fins al j. Els índexs es basen en 1. Per defecte, s'utilitza el primer caràcter.

```
code = ord("a")
```

sub

`sub(s, i, [j])`

Subcadena de i a j, base 1. Ometeu j per prendre la cua.

```
name = sub(keystr(), 1, 8)
```

split

`split(s, [sep, conv])`

Divideix s en una taula. sep per defecte és coma. Un nombre sep és l'amplada d'un bloc. conv converteix les peces amb tonum quan és cert.

```
words = split("red,blue", ",")
```

tonum

`tonum(s, [base])`

Analitza un nombre. La base pot ser de 2 a 36. Retorna zero quan el text no és un nombre.

```
n = tonum("255")
```

tostr

`tostr(v, [digits])`

Converteix un valor en text. dígits estableix decimals per a nombres.

```
print(tostr(hp, 0), 4, 4, 7)
```

Àudio

sfx

`sfx(n, [ch, offset, len])`

Reprodueix el patró n i retorna el canal. Si ometem ch, o passem -1, es tria el primer canal lliure. ch -2 atura el patró n a cada canal. n -1 atura un canal o tots els canals quan s'omet ch. n -2 permet que un patró en bucle acabi en comptes de repetir-se. sfx() sense arguments retorna -1.

```
ch = sfx(0)
```

sfxget

`sfxget(n, step)`

Retorna una taula d'un pas de patró: [1] to, [2] ona, [3] volum, [4] efecte. El to és de 0 a 63 (de C-0 a D#5). L'ona és de 0 a 15. El volum és de 0 a 7 i 0 és silenciós. L'efecte és de 0 a 7.

```
step = sfxget(0, 0)
pitch = step[1]
```

sfxset

`sfxset(n, step, pitch, wave, vol, fx)`

Escriu un pas de patró. Calen els sis arguments.

```
sfxset(0, 0, 24, 3, 5, 0)
```

sfxhdr

`sfxhdr(n, [key, val])`

Capçalera del patró. sfxhdr(n) retorna la taula de capçalera. sfxhdr(n, key, val) estableix un camp: speed, loop_in, loop_out, level, noise, crush, dtune, rverb, dampn, swing, steps. Els números del 0 al 10 seleccionen els mateixos camps en aquest ordre. speed és TIC, ticks per pas. Com més alt és més lent. 15 són 120 pulsacions per minut. El format de 16 bits accepta de l'1 al 255. Els sintetitzadors més estrets s'ajusten a la seva llista TIC.

```
sfxhdr(0, "speed", 15)
```

sfxstat

```
sfxstat([ch])
```

L'identificador del patró s'està reproduint al canal ch o -1. El canal per defecte és 0.

```
if sfxstat(0) < 0 then sfx(1) end
```

note

```
note(pitch, [wave, vol, fx, ch, ticks])
```

Reprodueix una nota sense patró. To 0..63, ona 0..15, volum 0..7, efecte 0..7. Els tics són el temps que es manté, en els mateixos tics de 120 Hz que TIC. Si ometeu els tics, dura 15, un pas per defecte. L'efecte s'executa durant aquest temps i després el canal es queda en silenci.

```
note(36, 3, 5, 5, 0)
```

music

```
music([n, fade, mask])
```

Comença la cançó n des de dalt. music(-1) s'atura. Si crides music amb l'ID que ja s'està reproduint, la deixes en funcionament. music() retorna l'ID de la cançó que es reprodueix, o -1. fade són mil·lisegons: la cançó entra o surt quan n és -1. Els bits de màscara que estan definits són els canals utilitzats per a la cançó. 0 significa que tots vuit. Els altres canals queden lliures per a sfx.

```
music(0, 1000, 7)
```

musicstat

`musicstat()`

Taula de la cançó que s'està reproduint. song i row són l'identificador de la cançó i la fila, o -1 quan no s'està reproduint res. step i pos tenen vuit números. step[1] és el canal 0, step[8] és el canal 7. step és l'índex de la nota, o -1 quan aquest canal està en silenci. pos és la distància que ha recorregut aquesta nota, de 0 a 1. tick és quants ticks ja s'ha reproduït aquesta fila. played és quantes files han començat des de music(), inclosa la que s'està reproduint.

```
st = musicstat()
row = st.row
note = st.step[1]
```

songget

`songget(song, [row, ch])`

L'identificador del patró s'emmagatzema en una cel·la de cançó. ch és 0..7. 255 significa que la cel·la està buida. Aquesta és la crida en directe; el text de la pista de codi encara mostra un ordre d'arguments més antic.

```
id = songget(0, 0, 0)
```

songset

`songset(song, row, ch, pattern)`

Escriu un ID de patró en una cel·la de cançó. Utilitzeu 255 per buidar la cel·la.

```
songset(0, 0, 0, 1)
```

stat

`stat(n)`

Només àudio. 16..23 és l'identificador d'efectes sonors al canal n-16. 24..31 és el patró musical al canal n-24. Qualsevol altre valor de n retorna 0. El temps de fotograma i la RAM són dades de rendiment i informació, no estadístiques.

```
id = stat(16)
```

Taules de dades

dget

`dget(t, row, col)`

Llegeix una cel·la de la taula de dades t del panell DADES. La fila i la columna comencen a 0.

```
hp = dget(0, 0, 1)
```

dset

`dset(t, row, col, v)`

Escriu una cel·la. Utilitza una cadena per a una columna de text.

```
dset(0, 0, 0, "slime")
```

dstr

`dstr(t, row, col)`

La cel·la com a text. Els nombres es converteixen.

```
name = dstr(0, row, 0)
```

drows

`drows(t)`

Quantes files té la taula t.

```
for r = 0, drows(0) - 1 do
end
```

dcols

`dcols(t)`

Quantes columnes té la taula t.

```
cols = dcols(0)
```

Desar i assoliments

save

`save(k, v)`

Emmagatzema v sota la clau k al fitxer de desament. v pot ser un número, una cadena o una taula d'aquests. El fitxer està comprimit.

```
save("hp", hp)
```

load

`load(k)`

Llegeix la clau k. Retorna zero quan falta la clau.

```
hp = load("hp") or 3
```

delsave

`delsave(k)`

Elimina la tecla k de la carpeta desada.

```
delsave("hp")
```

saveslot

`saveslot([name])`

Tria el fitxer desat dins la carpeta de l'autor. Sense nom, la ranura és el títol del joc. Retorna 1 quan aquest fitxer ja existeix.

```
if saveslot("slot1") == 1 then
  hp = load("hp") or 3
end
```

achieve

`achieve(i)`

Desbloqueja l'assoliment i, de 0 a 15. S'emmagatzema localment. Una compilació de Steam també estableix ACH_i. Una sessió de catàleg ho informa al compte del jugador.

```
if coins >= 100 then achieve(0) end
```

achieved

`achieved(i)`

1 quan l'assoliment i ja està desbloquejat.

```
if achieved(0) == 1 then spr(8, 4, 4) end
```

achdef

`achdef(i, text, [spr, tex])`

Text de brindis per l'assoliment i, que es mostra durant 15 segons des de la part inferior de la pantalla. spr és una fitxa opcional de 8×8, textualiza el seu full.

```
achdef(0, "rich", 8)
```

achclear

`achclear(i)`

Esborra l'assoliment i del fitxer local. Un ACH_i de Steam que ja estava definit es manté definit.

```
if btnd(5) then achclear(0) end
```

Xarxa

whoami

`whoami()`

El teu ID de jugador, 0..15, mentre hi ha una sessió activa. Fora de línia és -1.

```
id = whoami()
```

nhost

`nhost()`

1 si aquesta màquina és l'amfitrió, altrament 0.

```
if nhost() == 1 then spawn() end
```

nplayers

`nplayers()`

Quants identificadors en directe hi ha a 0..15.

```
print(nplayers(), 4, 4, 7)
```

nlisten

`nlisten()`

Inicia l'allotjament d'UDP i retorna l'adreça LAN com a ip:port.

```
addr = nlisten()
```

njoin

`njoin(addr)`

S'uneix a ip o ip:port.

```
njoin("10.0.0.8:7777")
```

nleave

`nleave()`

Surt de la sessió. Els iguals veuen _mp_leave amb el valor 0.

```
if btnd(11) then nleave() end
```

nsend

`nsend([p,] name, ...)`

Envia un missatge amb pèrdua. El nom és l'esdeveniment. Segueixen fins a quatre valors addicionals. p de 0 a 15 xiuxiueja a aquest equivalent; ometeu p per enviar-lo a tothom.

```
nsend("pos", x, y)
```

ncall

`ncall([p,] name, ...)`

Mateixa forma que nsend, però el missatge es torna a intentar fins que es confirma. Tres errors seguits maten la sessió.

```
ncall("hit", target, dmg)
```

nlan

`nlan()`

IP:port de la LAN actual després de nlisten.

```
print(nlan(), 2, 2, 7)
```

nstun

`nstun()`

El vostre port IP públic de STUN. Un convidat d'una altra xarxa pot intentar unir-s'hi.

```
pub = nstun()
```

setturn

`setturn(addr, [user, pass])`

TURN fallback quan falla una ruta directa. setturn() amb arguments buits la desactiva. L'adreça és host:port.

```
setturn("turn.example:3478", user, pass)
```

nlist

`nlist(game)`

Taula de sessions obertes per a aquest nom de joc. Cada entrada té una adreça, jugadors i nom.

```
rooms = nlist("duel")
```

nannounce

`nannounce(game)`

Batec de cor perquè aquest amfitrió aparegui a nlist per a la partida. Crida'l amb un temporitzador mentre allotja.

```
nannounce("duel")
```

nkb

`nkb([on])`

Camp d'adreça. nkb(1) l'obre, nkb(0) l'amaga, nkb() t'indica si està obert. Amunt i avall canvien un caràcter, del 0 al 9 o un punt o dos punts. A ho escriu. X esborra. El teclat no hi escriu.

```
nkb(1)
```

nkbstr

`nkbstr([s])`

Text al camp d'adreça, i després els caràcters amunt i avall canvien. nkbstr("") esborra el text.

```
addr, digit = nkbstr()
```

_mp

`_mp(from, name, ...)`

Escriviu aquesta funció. S'executa quan arriba un missatge. "from" és l'ID del remitent. "name" és l'esdeveniment. A continuació, hi ha un màxim de quatre valors.

```
function _mp(from, name, a, b)
  if name == "pos" then
    px[from] = a
    py[from] = b
  end
end
```

_mp_join

`_mp_join(p)`

Escriviu aquesta funció. Un node ha rebut l'id p. També s'activa per a vosaltres; whoami() és aquest id.

```
function _mp_join(p)
  add(log, p)
end
```

_mp_leave

`_mp_leave(p, why)`

Escriu aquesta funció. El peer p va marxar. Per què 0 és una sortida normal. Per què 1 és una connexió perduda.

```
function _mp_leave(p, why)
  px[p] = nil
end
```

_mp_update

`_mp_update()`

Escriu aquesta funció. S'executa cada 15 fotogrames d'actualització, uns 4 Hz. Un bon lloc per enviar posicions sense enviar.

```
function _mp_update()
  nsend("pos", x, y)
end
```

_mp_fail

`_mp_fail()`

Escriviu aquesta funció. La sessió ha mort: l'amfitrió ha desaparegut o l'ncall ha fallat tres vegades.

```
function _mp_fail()
  online = false
end
```

Coroutines

cocreate

`cocreate(f)`

Crea una corrutina a partir de la funció f. No s'executa fins que no es realitza la coresume.

```
co = cocreate(cutscene)
```

coresume

`coresume(c, ...)`

Executa la coroutina c fins que retorna un valor. En cas d'èxit: true i els valors obtinguts. En cas d'error: false i l'error.

```
ok, a = coresume(co)
```

costatus

`costatus(c)`

"suspès", "corrent" o "mort".

```
if costatus(co) == "dead" then co = nil end
```

yield

`yield(...)`

Retorna el control a qui hagi cridat coresume. Els valors esdevenen els resultats d'aquest coresume.

```
yield(1)
```

Mesura

info

```
info([x, y, col])
```

Dibuixa el HUD en directe a _draw: fps (mitjana d'1 segon), upd, drw, px, max, ram, peak, ramc i net quan una sessió està activada. info() utilitza el cursor d'impressió. info(col) només estableix el color. info(x, y) o info(x, y, col) el situen. Els números utilitzen B, k o m perquè s'ajustin a una pantalla petita. La trucada en si no es compta a px.

```
info(1, 1, 7)
```

showstats

```
showstats([on])
```

El mateix HUD que la informació, dibuixat automàticament a 0, 0 després de _draw, ignorant la càmera i el clip. showstats(0) ho amaga. showstats(1) o showstats() ho mostra.

```
showstats(1)
```

perf

```
perf(n)
```

Un número de HUD, perquè el puguis dibuixar tu mateix. 0 fps, 1 actualització, 2 dibuixos, 3 px, 4 màxim (el fotograma més pesat d'aquest segon), 5 RAM, 6 RAM (RAM més el cartutx de joc empaquetat), 7 de RAM màxima des que va començar el cartutx de joc.

```
print(perf(0), 2, 2, 7)
```