PIX37 kézikönyv

Szerkesztő, majd programozás. A könyv offline példánya elérhető a pix37.party/docs oldalon.

Szerkesztő

A PIX37 DEV a szerkesztő. Itt rajzolhatsz, írhatsz és meghallgathatod a játékot, majd exportálhatsz egy játékkazettát vagy egy önálló buildet. Le is tud játszani egy játékkazettával, és megnyithatja a PIX37.PARTY fájlt a játékkazetták letöltéséhez és a katalógus használatához. A PIX37 DEV EDUCATIONAL ugyanaz a szerkesztő, és ingyenes. Nincsenek buildjei, nincs játékkazetta-exportja, és nincs hozzáférése a PIX37.PARTY-hoz. A PIX37 Console a lejátszó. Lejátssza a `.p37` játékkazettákat, és megnyithatod a PIX37.PARTY fájlt a játékkazetták letöltéséhez és a katalógus böngészéséhez.

Projektek és játékkazetták

A `.pix37` fájl a projekt: forráskód fülek, sprite lapok, térképek, hangok és adattáblák. A `.p37` fájl a játékos által futtatott játékkazetta: szorosan csomagolt bájtkód.

A mentések és az eredmények a szerző mappában találhatók az alkalmazásadatok mellett. Windows rendszeren ez `C:\\Users\\Ada\\AppData\\Roaming\\PIX37\\Mira\\Pong.p37s` mentés esetén, és `C:\\Users\\Ada\\AppData\\Roaming\\PIX37\\Mira\\Pong.ach` eredmények esetén. Linux rendszeren ugyanezek a fájlok a `~/.local/share/PIX37/Mira/Pong.p37s` és a `~/.local/share/PIX37/Mira/Pong.ach`. Az Ada a számítógépen lévő fiók. A Mira a szerző neve, és ez a név a mappa. A Pong a mentési hely, amely alapértelmezés szerint a játék címét írja. A `save` a `.p37s` fájlt írja. Az első feloldott eredmény a `.ach` fájlt írja. Ha a fájlok már ott vannak, a futtató beolvassa őket, amikor a játék kazetta elindul.

Fejlesztői menü

Az Escape megnyitja a fejlesztői menüt. Játék futása közben az Escape gombbal visszatérhet a szerkesztőbe. Erősítse meg az Enterrel vagy az A gombbal (alapértelmezés szerint V). A Vissza gomb a B gomb (alapértelmezés szerint C).

Ha nincs betöltve projekt, a menü a következő: PROJEKT LÉTREHOZÁSA, PROJEKT BETÖLTÉSE, ÉPÍTÉS, Játékkazetta BETÖLTÉSE, CSATLAKOZÁS A PIX37.PARTY-HOZ, BEÁLLÍTÁSOK.

Miután betöltődött egy projekt, a következőket kell tennie: PROJEKT MENTÉSE, PROJEKT BETÖLTÉSE, PROJEKT LÉTREHOZÁSA, ESZKÖZÖK, ÉPÍTÉS, Játékkazetta BETÖLTÉSE, CSATLAKOZÁS A PIX37.PARTY-HOZ, BEÁLLÍTÁSOK.

A Ctrl+S billentyűkombináció menti a megnyitott projektet. A Ctrl+R billentyűkombináció futtatja.

Projekt létrehozása

A CREATE PROJECT parancs a projekt elkészítésének megkezdése előtt kéri a játék kazettáját.

- **VIDEÓ** — 1 BITES (2 szín), 2 BITES (4), 4 BITES (16), 8 BITES (256).
- **HANG** — a szintetizátor mintavételezési mélysége: 16, 8, 4, 2 vagy 1 bit.
- **KÉPERNYŐ** – az előre beállított méretek közé tartozik a 64×64, 128×128, 160×90, 256×256 és 320×180. Az EGYEDI méret lehetővé teszi, hogy 1×1 és 512×512 között bármilyen méretet válasszon.
- **PALETTÁK** — színbankok. Az első bank az alábbi 16 színű készlettel indul. A PALETTA BETÖLTÉSE egy beépített készletet alkalmaz: PICO-8 16, PICO-8 32, Game Boy, 1-bit, CGA, Sweetie-16, és egy hosszabb lista a Lospec-ből (Resurrect 64, Apollo, Endesga 32 és 64, CC-29, SLSO8, Pear36, Oil 6, Duel, AAP-64, Lost Century és mások). Betölthetsz egy palettafájlt is, vagy betöltheted a saját PNG-det, és a palettát ebből a képből veheted ki.



A `cls`, `pset`, `spr` és a többi által használt színindexek. Bármelyik indexet átfestheted a palettaszerkesztőben.

Az öt panel

A szerkesztő a GAME, PAINT, CODE, SOUND és DATA, ebben a sorrendben. A térkép a PAINT-ben található. A hangeffektek és a dalok a SOUND-ban találhatók. Kattints egy panelre a munkához. Amikor a CODE van fókuszban, a SOUND és a DATA figyelmen kívül hagyja a betűbillentyűket, így a gépelés nem egyben a zongorázást is jelenti.



Teljes elrendezés, `1:1:4:1:1` betűtípussal. Minden szám azt jelzi, hogy az adott panel mekkora részét foglalja el a sornak: GAME, PAINT, CODE, SOUND, DATA. Egy kis képernyő egy 128×128-as cella közepén helyezkedik el. A 128 képpontnál nagyobb képernyő kiszélesíti a GAME oszlopot.

BEÁLLÍTÁSOK → GLOBÁLIS → ELRENDEZÉS menüpontban válthat az elrendezések között:

Kód	Amit látsz
`1:1:4:1:1`	Minden. JÁTÉK és HANG egymásra rakva, FESTÉS és ADATOK egymásra rakva, KÓD jobbra.
`1:1:4:1:1 2X`	Ugyanaz az elrendezés. A KÓD kétszeres sűrűséggel van kirajzolva, így kétszer annyi szöveg fér bele.
`1:0:1:0:1`	JÁTÉK, ADATOK, KÓD.
`1:0:1:0:1 2X`	Ugyanez az elrendezés, csak a sűrűbb KÓDDAL.
`0:1:1:1:1`	Nincs játékelőzetes. FESTÉS, KÓD, HANG, ADATOK.
`0:1:1:1:1 2X`	Ugyanez az elrendezés, csak a sűrűbb KÓDDAL.
`0:1:4:1:2`	Stúdió. Széles KÓD, egy magasabb DATA szalag, FESTÉS és HANG.
`0:1:4:1:2 2X`	Ugyanez az elrendezés, csak a sűrűbb KÓDDAL.

`0:0:4:0:4`	Csak KÓD és ADAT, mindkettő nagy.
`0:0:4:0:4 2X`	Ugyanez az elrendezés, csak a sűrűbb KÓDDAL.
`0:1:4:1:0`	FESTÉS, HANG és széles KÓD.
`0:1:4:1:0 2X`	Ugyanez az elrendezés, csak a sűrűbb KÓDDAL.
`0:4:0:0:0`	Csak FESTÉS, 256×256, a lap egy pixele a vászon egy pixeléhez viszonyítva.

Egy `2X` sor megtartja ugyanazokat a paneleket. A CODE panel kétszeres felbontással jelenik meg, és ugyanabba a cellába van visszaméretezve, így a szöveg, a tabulátorok és a kötésmargó fele olyan magas, és kétszer annyi forrás fér el a képernyőn.

JÁTÉK

A futó játékkazetta előnézete. Kattintson a JÁTÉK panelre, vagy nyomja meg a Ctrl+R billentyűkombinációt az előnézet futtatásához. A BEÁLLÍTÁSOK → JÁTÉK PANEL menüpontban konfigurálhatja a panelt:

- **Előnézet** – tétlen színes, folyamatos vagy automatikus újratöltés szerkesztéskor.
- **Új ablakban játék** — a játék a szerkesztő mellett nyílik meg.
- **Kijelző** – előnézet RGB888 vagy RGB565 formátumban.
- **GAME 64** – egy 64×64-es képernyő vagy 1:1 arányban van középre igazítva a cellában, vagy 2×-es méretarányban van kitöltve.
- **Shaderek** – CRT és szűrők a játékban, elkülönítve a szerkesztő szűrőitől.
- **Szín** – a panel háttere.

FESTÉK

Sprite lapok és ugyanazon panel térkép oldalán csempézett térképek. A lapok nagyobbak lehetnek, mint a képernyő. A sprite előnézeti méretei 8, 16, 32, 64 és 128 (128 csak akkor, ha a lap legalább 128 képpontos). A nagyításhoz görgessen az egérgörgővel a sprite előnézet fölé. Az ecsetméretek 1, 2, 3 és 4.

Kulcsfontosságú	Eszköz
B.	Ceruza
M	Téglalap kijelölése
Q	Lasszó
S	Bélyeg

G	Vödör
én	Szemcseppentő
V.	Mozog
L / Shift+L	Vonal / görbe
U / Shift+U	Üreges téglalap / üreges ellipszis
F / Shift+F	Kitöltött téglalap / kitöltött ellipszis
D	Remeg
[ ]	Kisebb / nagyobb ecset
Ctrl+Z / Ctrl+Y	Visszavonás / Újra végrehajtás. A Ctrl+Shift+Z billentyűkombinációval szintén visszavonható.
Ctrl+C / Ctrl+X / Ctrl+V	Másolás, kivágás, beillesztés
Ctrl+A	Jelöld ki a vásznat

A térkép ugyanazt az elvet használja, kevesebb eszközzel: B ceruza, M téglalap kijelölés, S bélyegző, V mozgatás, valamint Ctrl+Z, Ctrl+Y, Ctrl+C, Ctrl+V. Az Alt billentyűt lenyomva tartva nagyíthatod a térkép textúráját. A térképen az ecset mérete a festőeszközökkel megegyező marad.

Minden sprite nyolc flag bitet tartalmaz. A Paint program beállítja ezeket a sprite-on; a `fget` és a `fset` program olvassa be őket a játékban.

BEÁLLÍTÁSOK → FESTÉSSZERKESZTŐ menüpontban konfigurálható a panel:

- **Paletta** — melyik bank kerül kihúzásra, plusz betöltés és mentés.
- **Előnézet** — RGB888 vagy RGB565.
- **Ecsetkurzor** — a mutató festés közben.
- **Pixel snap** — hogyan illeszti az ecset a pixeleket a képpontokhoz.
- **Nagy ecset** – egy nagy ecset a bal felső sarkából vagy a teljes alapterületéről pattintható ki.
- **Hagymahéj** — más keretek szelleme, és hogy melyik irányba néz az a szellem.
- **Térképnagyítás** — valódi pixelek, vagy szabad nagyítás.
- **Színek** — a kezelőfelület színei.

KÓD

A tabok különálló fájlok a projekten belül. Kattintson egy lap nevére az átnevezéshez; az Enter billentyűvel megerősítheti. Több lap, amelyek mindegyike `_init`, `_update` vagy `_draw`, tabulátorsorrendben van

összefűzve: az első tabulátor fut először, az utolsó tabulátor utoljára. A hookokon belül nem lévő kód a `_init` részévé válik. A hookok nélküli játékkazettákat egyetlen `_draw`-ként kezeli a rendszer.

Kulcsfontosságú	Akció
Nyilak	A kurzor mozgatása. A Shift billentyűvel kiterjeszthető a kijelölt terület.
Ctrl+Balra / Jobbra	Szó szerint.
Home / End, Ctrl+Home / End	A sor eleje vagy vége, vagy a tabulátorjel.
Fül	Szúrjon be két szóközt. A kiegészítési listában fogadja el a nevet.
Belépés	Új sor, behúzva, mint az előző. Elfogadja a kiegészítést.
Ctrl+C / X / V	Másolás, kivágás, beillesztés.
Ctrl+Z / Ctrl+Y	Visszavonás / újraindítás.
Ctrl+A	Válassza ki a fület.
Ctrl+D	A vonal vagy a kijelölés duplikálása lefelé.
Ctrl+/	Hozzászólás írása vagy megjegyzés törlése a `--` paranccsal.
Ctrl+F	Keresés. Az F3 a következő találatra ugrik. Az Escape bezárja a sávot.
Ctrl+G	Ugrás egy sorszámra.
Menekülés	Először bezárja a keresést, az ugrást vagy a tipplistát. A második Escape billentyűvel elhagyhatja a szerkesztőt, és a fejlesztői menübe juthat.

Függvénynév beírása közben tippek jelennek meg. Az Enter vagy a Tab billentyű beszúrja a nevet, és egy `(` kódot, ha a COMPLETE (be van kapcsolva a beállításokban. BEÁLLÍTÁSOK → KÓDSZERKESZTŐ:

- **Keresés** – Ctrl+F az aktív lapon, vagy az összes lapon.
- **Magas szöveg** – a karakterjelek levágása 5 képpontra, vagy a teljes karakterjel megjelenítése.
- **Karakterlánc mérete** — a karakterlánc literálok betűmérete.
- **Megjegyzés betűtípus** — 3×5, vagy az aktuális felhasználói felület nyelvének Fusion betűtípusa.
- **Befejezett (** — hogy a befejezés típusa legyen-e a zárójel.
- **Kódtippek** – be vagy ki.
- **Tippek nyelve** — a menü nyelve, vagyis angol.
- **Kurzorköz** – extra eltolás az üres sorokon.
- **Ereszcsatorna** – sorszámok.
- **Szín** – háttér, megjegyzések, tabulátorok, számok, operátorok, kulcsszavak, karakterláncok, motorfüggvények, a saját függvényeid, változók, zárójelek, behúzott szegély, kijelölés, hibasor, figyelmeztetés

HANG

A panelnek két füle van. Az SFX egy pattern: egy rövid hangjegysorozat. A SOUND egy dal: ezeket a pattern-eket nyolc csatornára helyezi el. A TIC azt jelzi, hogy egy lépés hány tiketet vár. Az órajel másodpercenként 120 tiket, és egy lépés egy tizenhatodik hang, tehát 15 tiket 120 ütés percenként. A magasabb TIC lassabb pattern. Két pattern lépésben marad, amikor az egyik TIC elválasztja a másikat: a 15 és a 30, a 15 és a 8 nem.

A zongora akkor szólal meg, amikor egy hangot kijelölünk az egérrel. A fehér billentyűk ZXCVBNM (CDEFGAB). A fekete billentyűk SDGHJ (C# D# F# G# A#). Vigye az egérmutatót egy érték fölé, és írjon be egy számot, vagy vigye az egérmutatót rá, és görgesse a kereket. A kódban a hangmagasság-számok 0-tól (C-0) 63-ig (D#5) terjednek.

Az egyik pattern-en a LOOP és a LEN ugyanaz a két szám. A 0 feletti OUT IN-től OUT-ig ismétlődik, és a cím LOOP. A 0-nál lévő OUT és a 0 feletti IN a LEN: ennyi hang szólal meg az elejétől, majd a pattern véget ér, és a cím LEN. Mindkettő 0-nál minden lépést lejátszik.

A szintetizátor szerkeszthetősége a játék kazettájához kiválasztott bitmélységtől függ. BEÁLLÍTÁSOK → SFX SZERKESZTŐ állítja be ezt a mélységet, azt, hogy egy hangjegy automatikusan továbblépjen-e (ki, hangjegylánc vagy mezőlánc), hogy a szóköz újraindítja-e a lejátszást, vagy ki-be kapcsolja az indítást és a leállítást, valamint a szerkesztő színeit.

16 bites

Minden lépéshez tartozik egy hangjegy, egy hangerő, egy hullámforma és egy effekt. A hangerő 0 és 7 között van. A 0 néma.

Hullámok:

- **0 háromszög**
- **1 döntött fűrész**
- **2 fűrész**
- **3 négyzet**
- **4 pulzus** – keskeny pulzus
- **5 szerv**
- **6 zaj**
- **7 fázisú**
- **8 PWM** — egy négyzet, amelynek szélessége mozog
- **9 impulzus 25%**
- **10 szinusz**
- **11 fémes zaj**
- **12 FM csengő**
- **13 pengetés**
- **14 szuperfűrész**

- **15 PWM (C64)**

Effektek egyetlen hangjegyre:

- **0 nincs**
- **1 csúsztatás** – a hangmagasság és a hangerő a következő hangjegyre csúszik ebben a lépésben
- **2 vibrato** — a hangmagasság ingadozik
- **3 csepp** – a hangmagasság elcsendesedik, a szokásos rúgás
- **4. elhalványulás**
- **5 elhalványul**
- **6 arpeggio** – ezt a hangot és a következő hármat 8 tikkenként lépteti. 8-as vagy az alatti hangfekvésen 4 tikkenként.
- **7 gyors arpeggio** – ugyanaz a négy hangjegy, minden 4 tiketten, vagy minden 2-en, ha a TIC 8 vagy az alatt van.

A minta fejléce, a jegyzetek felett:

- **TIC** — ütemek száma lépésenként, 1-től 255-ig. A 15 percenként 120 ütést jelent. Írd be a számot, vagy léptesd eggyel.
- **LOOP IN / OUT** – ha az OUT érték nagyobb, mint 0, a pattern IN-től OUT-ig ismétlődik. Hagyjuk az OUT értéket 0-n, és a cím LEN lesz: az IN a lejátszott hangok számát jelöli, a 0 pedig a teljes pattern-t.
- **ZAJ** — az egész minta zajjá válik.
- **CRUSH** — az egyes hullámok egy vibráló változata.
- **DTUNE** — 0 kikapcsolva, 1 egy halk kórus, 2 egy erős kórus.
- **RVERB** — zengetés: 0 kikapcsolva, 1 rövid, 2 hosszú.
- **DAMPN** — aluláteresztő szűrő: 0 kikapcsolva, 1 gyenge, 2 erős.
- **SZINT** — a teljes minta hangereje, 0-tól 7-ig.
- **HINTA** — csoszog. A szokatlan lépések kicsit tovább várnak, az egyenletesek kicsit kevesebbet.
- **LÉPÉSEK** — a minta lépésszáma: 32, 24, 16 vagy 12.

8 bites

Egy lépéshez tartozik egy hangjegy és egy hullámforma. A négy hullámforma a négyzet, a háromszög, a fűrész és a zaj. A fejlécben található a TIC, a LOOP IN / OUT, az FX és a LEVEL. A TIC a következők egyike: 1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 16, 20, 24, 30, 40, 60. Az FX egy effekt a teljes patternre: 0 nincs, 1 vibrato, 2 arpeggio, 3 fade out. A LEVEL 0-tól 3-ig terjed.

4 bites

Egy lépés csak egy hangjegy. A fejlécben található a TIC, LOOP (be vagy ki), WAVE (ugyanaz a négy hullám) és OCT (melyik oktávban helyezkednek el a hangjegyek). A TIC a következők egyike: 4, 6, 8, 10, 12, 15, 20, 30.

2 bites

Egy lépés üres, egy dob, egy ütés vagy egy zúzás. A fejlécben van TIC és BASE. A TIC értéke 8, 15, 16 vagy 30. A BASE beállítja, hogy milyen magasan legyen a dob és az ütés.

1 bites

Egy lépés egy hangjegy. A TIC értéke 8, 15, 16 vagy 30, ugyanaz a négy sebesség, mint a 2 bitesnél.

Dalok

A HANG fül a megnyitott dalt jelöli. Nyolc oszlop a csatornákat jelöli. Egy cella egy minta azonosító. A 255-ös, üresen rajzolt érték azt jelenti, hogy a csatorna néma az adott sorban. Az X oszlop egy parancs: folytatás, leállítás, ciklus indítása vagy ciklus vége. A Stop a sor után leállítja a dalt. A ciklus vége visszaugrik az utolsó ciklus elejére.

Egy sor akkor ér véget, amikor a bal szélső, 0-nál lévő LOOP OUT minta véget ér. A jobbra lévő minták nem hosszabbítják meg a sort: egy hosszabb elvágásra kerül, egy rövidebb pedig az első lépéstől kezdődik újra, amíg a sor véget nem ér. Egy 0 feletti LOOP OUT minta a LOOP IN és a LOOP OUT között ismétlődik, és nem fejezi be a sort. Ha a sorban minden minta ismétlődik, a sor ott marad.

Gyorsbillentyűk

Kulcsfontosságú	Akció
Zongorabillentyűk	Írjon jegyzetet az egérrel kiválasztott lépéshez.
Fel / Le	Módosítsa a kurzor alatti értéket.
Tér	Játssza le a nyitott SFX-et, vagy az első sor nyitott dalát. Indítás/leállítás vagy újraindítás az SFX SZERKESZTŐ beállításainál.
Ctrl+Z / Ctrl+Y	Visszavonás / újraindítás.
Ctrl+C / Ctrl+X / Ctrl+V	A kijelölt rész másolása, kivágása, beillesztése.

ADAT

Elnevezett táblák. Minden tábla sorokból és oszlopokból áll. A kódból a `dget`, `dset`, `dstr`, `drows` és `dcols` használja ezeket a táblákat. A sor- és oszlopszámozás a kódban 0-val kezdődik.

Oszloptípusok, ismételve a fejlécben:

- **bool** — egy bit.
- **2** — két bit.
- **4** — négy bit.

- **8** — egy bájt.
- **16** — 16 bit, azonosítókhoz és nagyobb számokhoz.
- **spr** — egy sprite azonosító.
- **str** — szöveg, a rendszer betűtípusával kiírva.
- **str8** és **str10** – ugyanaz a szöveg, csak magasabbra rajzolva.
- **f16** és **f32** – félpontosságú és teljes pontosságú lebegőpontos.

Ha a mutató egy oszlopfejlécen van, a Ctrl+görgetés billentyűkombinációval megváltoztatható az adott oszlop típusa. Egy sima görgetés oszlopok között vagy sorok alatt mozog, attól függően, hogy a BEÁLLÍTÁSOK → ADATVÉDELMI SZERKESZTŐ menüpontban melyiket választjuk. A Ctrl megfordítja ezt az irányt, ezért a Ctrl+görgetés egy fejlécen megváltoztatja a típust, amikor a görgő oszlopokra van állítva.

Nyílbillentyűk mozognak. Az Enter szerkeszti a cellát. BEÁLLÍTÁSOK → ADATVÉDELMI SZERKESZTŐ:

- **Görgő** – oszlopok vagy sorok görgetése.
- **Színek** – a rács, a fejléc és a kijelölés.

Eszközök

Az ASSETS parancs importálja és exportálja a megnyitott darabot, nem az egész projektet. A sprite-ok a PAINT-ben, a megnyitott munkalapon megnyitott sprite-ot használják. A térkép a megnyitott térkép. Az SFX a megnyitott minta. A zene a megnyitott dal. Az adat a megnyitott táblázat. A kód a kivétel: minden tabulátor kiíródik, és minden tabulátor visszaolvasódik.

Fajta	Fájl	Mi mozog
Sprite-ok	`.png`	A nyitott sprite.
Térkép	`.p37map`	A nyitott térkép.
Hangeffektek	`.p37sfx`	A nyitott minta.
Zene	`.p37mus`	A nyitott dal. A pattern számok változatlanok maradnak. Exportáld a hangeffekteket külön, ha szükséged van ezekre a hangokra.
Adat	`.p37dat`	A nyitott asztal.
Kód	`.lua`	Minden lap.

Beállítások

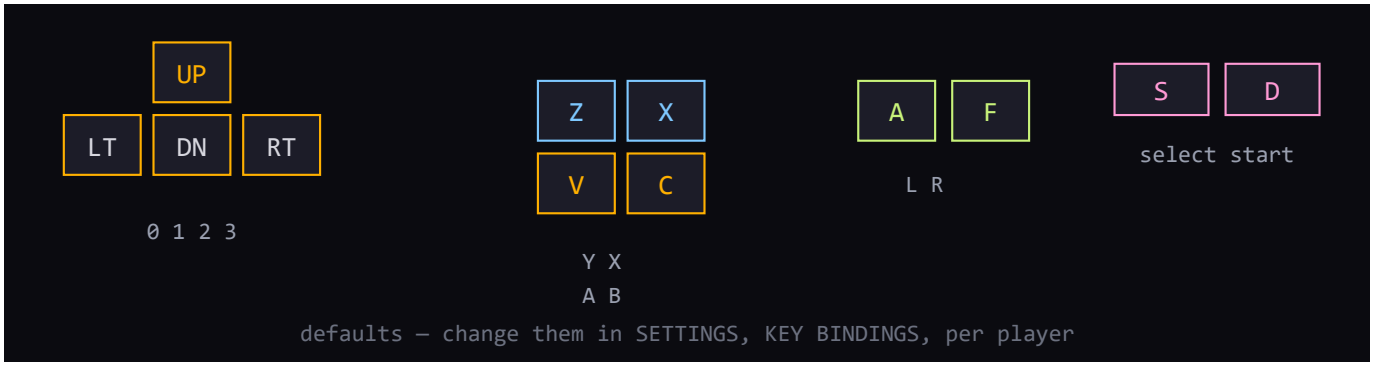
A gyökérlista:

Oldal	Mit tárol

PROFILOK	Mentse el és töltse be az összes beállítást. Új, átnevezés, felülírás, törlés. Profil másolása vagy beillesztése szövegként.
Menü	Ablak, billentyűk, menüszínek, opcionális háttér.
JÁTÉKPANEL	Előnézet és a lejátszási ablak.
KÓDSZERKESZTŐ	Kiemelés és tippek.
FESTÉSSZERKESZTŐ	Ecset, paletta, térképnézet.
SFX SZERKESZTŐ	Szintetizátor és hangjegy bevitel.
ADATVÉDELMI SZERKESZTŐ	Kerekek iránya és színei.
NYELV	Menü és tippek nyelve.
ENGEDÉLYEK	Licencszövegek, beleértve a Fusion Pixel betűtípust és az SDL3-at.

GLOBÁLIS:

- **ELRENDEZÉS** — panelelrendezés, fent.
- **KÖNNYŰSÉG** — mesterfok, 0-tól 10-ig.
- **BILLENTYŰKÖDÉSEK** — billentyűk mind a 8 játékoshoz. Nyomj meg egy billentyűt a lefoglalásához. Útmutató, ABXYLR, kiválasztás.
- **Jelen** — üres képkockák kihagyása, vagy minden képkocka kirajzolása.
- **Teljes képernyős** — szegély nélküli.
- **Frissítés** — ablak FPS. Balra és jobbra lépés: 60, 90, 120 és max. Írjon be egy számot 1000-ig. A 0 azt jelenti, hogy nincs felső határ.
- **Vsync** — várakozás a képkockák közötti megjelenítésre.
- **Skálázás** — egész ablakszorzó.
- **Shaderek** – CRT és szűrők a szerkesztőben, nem a játékban.
- **Szín** – menüszöveg és kiemelés.
- **Háttér** – egy opcionális PNG, 128-asra vágva, a menü mögött. A Törlés gombbal eltávolítható.
- **Szerkesztőkurzor** — mutató alakja.
- **Egér** – lebegő mutató, vagy képpontokhoz igazított.
- **Felhasználói felület sebessége** – milyen gyorsan ismétlődik egy megtartott irány egy listában.
- **Intéző** – a fájllisták a számítógép gyökérkönyvtárában, vagy az exe fájl mappájában kezdődnek.



Alapértelmezett billentyűzet. Az A a menükben a megerősítést jelenti. A B a visszatérést. Egy játékban ezek a 4-es és 5-ös gombok. A Select (S) a 10-es gomb, nem egy menüparancs.

Gomb	Szerep	Alapértelmezett kulcs
0	Balra	Balra mutató nyíl
1	Jobbra	Jobbra mutató nyíl
2	Fel	Felfelé mutató nyíl
3	Le	Lefelé mutató nyíl
4	A, erősítse meg	V.
5	B, hátul	C
6	X	X
7	Y	Z
8	R	F
9	L	Egy
10	Válasszon	S
11	Indul	D

Épít

Válassz platformot: Windows, Linux, Mac, Android, iOS vagy Web.

- **Név** — a build fájlneve.
- **Szerző** – Windows és Linux rendszeren. Ez a mentések mappaneve.
- **Steam alkalmazásazonosító** – Windows és Linux rendszeren. Az üres érték azt jelenti, hogy nincs Steamworks. A kitöltött azonosító a `achieve(0)`…`achieve(15)` karaktereket a Steam ACH_0…ACH_15

teljesítményeihez rendeli.
- **Id** — csomagazonosító, ahol a platform kér egyet.
- **Kód elrejtése** — asztali és mobilos átfedéseken. Lassítja a játékkazettát. `peek`. Ez nem zár.
- **Teljes képernyő** – az exportált lejátszó teljes képernyőn indul el.
- **Shaderek** – akár három réteg is beépítve a lejátszóba.
- **.p37 letöltése** – Csak webes használatra. Letöltési bejegyzést jelenít meg a szüneteltetés menüben.
- **Ikon** — egy sprite lapról származó csempe: lap, méret, x, y. Vagy egy PNG borító, amit kivágsz.
- **Exportálás** — válassza ki a mappát. A menüben a BUILD SAVED (ÉPÍTÉS MENTVE) vagy a BUILD FAILED (ÉPÍTÉS SIKERTELEN) felirat jelenik meg.

Készíts egy buildet bekapcsolt internettel. A splash névtelenül megkérdezheti a szervertől, hogy ismert-e a build hash-e. Ez az ellenőrzés két okból van ott. Megvéd engem, a PIX37 szerzőjét, az eszköz olyan másolataitól, amelyek nem az enyémek, hogy elajándékozzam őket. Védi a játékot készítő személyt is. Egyre gyakoribb a telefonpiacon, például a Play (Android) vagy az iOS rendszeren, hogy valaki más játékát sajátként listázd. A játék továbbra is elindul. Ha a gép offline állapotban van, az intró azt mondja, hogy a játék nincs csatlakoztatva. Ez a sor egyben utalás a játékosnak arra is, hogy a másolat nem feltétlenül legális forrásból származik. Ez egy apró ellenőrzés, így ha a játékodat ellopták és a beleegyezésed nélkül eladták valaki mástól, akkor ezt egy áruházban, például a Play (Android), a Steam vagy az itch.io áruházban is megteheted. A szerver a hash első regisztrációjának időpontját a hash mellett tárolja, és később nem módosítja. A korábbi időpont az eredeti build. Az ezután regisztrált hash egy későbbi másolat.

Azt akarom, hogy ez az ellenőrzés megvédje a játékfejlesztőket. A motor már elvégzi a munka nagy részét helyettük, és ez is egy ilyen dolog. Megértem, hogy ez ellentmondásos lehet. Ha a közösség úgy dönt, hogy jobb, ha elhagyjuk ezt a mikro-DRM-et, akkor megteszem.

Egy korábbi szerkesztőből származó régi játékkazetta megtagadhatja a rendszerindítást a nyelvváltás után. Nyisd meg a `.pix37` projektet ebben a szerkesztőben, és exportálj egy új `.p37` fájlt.

Csatlakozás a PIX37.PARTY-hoz

A PIX37.PARTY-HOZ KAPCSOLÓDÁS az alkalmazáson belül található. Nincs bejelentkezési oldal a böngészőben. Steam munkamenetjegyével vagy az itch.io-val kell bejelentkezni a pad és az egér használatával. A szolgáltatás egy 2-16 karakterből álló, egyedi megjelenített nevet jelenít meg. A licenckulcs és a Steam azonosító nem jelenik meg a többi játékos számára.

A szolgáltatási feltételek után a központ a Kedvencek, Katalógus, Rangsor, Játékosok és a profilod.

A Fejlesztői licenc a szerkesztő. Te készíted a játékot, feltöltöd, és csillagokat hagysz magad után. A Játékos licenc a játékos. A szíveket a futókonzolon játszó emberek adják.

- **Katalógus** – játékok böngészése. A szűrők a műfaj (legfeljebb három), a képernyőméret, a játékkazetta mérete, valamint az alapján készülhetnek, hogy a feltöltött anyag tartalmaz-e `.pix37` projektet.
- **Kedvencek** – megjelölt játékok.

- **Feltöltés** – fejlesztői licenc szükséges. Elküldsz egy `.p37` fájlt, és csatolhatod a `.pix37` projektet. A `.p37` csomagolt bájtkód. Nehéz olvasni, és szándékosan is összezavarható. A `.pix37` az a projektfájl, amelyben ténylegesen dolgozol: a forráskód, a képek és a hang, olyan formában, amelyet egy másik személy megnyithat és tanulhat belőle. Az elküldésével más alkotók tanulmányozhatják a játékot egy tömörített játékkazetta helyett. A cím, a szerző, a műfajok, a borító, az ikon és az eredmények nevei is megjelennek. A felküldött RAM szám a játékkazetta által elért csúcsérték.
- **Hozzászólások és eredmények** – a játék oldalán. Ha bejelentkezett állapotban van, egy eredmény feloldása az adott fiókban és a gépen is tárolódik.
- **Rangsor** – fejlesztők, csillagok vagy szívek szerint rendezve. Minden sor mindkét számot mutatja. Nyisson meg egy nevet, hogy megtekinthesse az adott szerző játékait szívek szerint rendezve.
- **Játékosok** — a feloldott eredmények szerint rendezett fiókok.

Kód

A nyelv egy kis Lua stílusú szkript. Ez a rész egy üres projektből indul ki, megtanítja a nyelvet, majd minden függvényre mutat egy rövid példát.

Egy első program

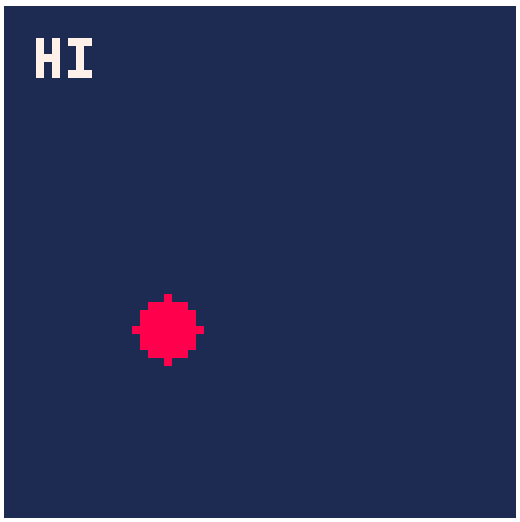
Hozz létre egy 64×64-es, 4 bites színmélységű projektet. Nyisd meg a CODE-ot. Cseréld le a tabulátort erre:

```
_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
```

A Ctrl+R billentyűkombinációval futtatható. A nyílbillentyűk mozgatják a kört. Az Escape leállítja a játékot és visszatér a szerkesztőbe.



Ugyanaz a három hívás: a `cls(1)` a sötétkék képernyőt, a `print` a HI-t, a `circfill` pedig a rózsaszín kört rajzolja.

Mi fut, és mikor

- A `_init` egyszer fut.
- A `_update` 60 Hz-en működik. Ide kell helyezni a mozgást és a gombokat.
- A `_draw` festést végez. FPS argumentum nélkül olyan gyorsan fest, ahogy a számítógép és a monitor bírja. A `_draw(30)` a festést 30 fps-re korlátozza, a logikát pedig 60-ra hagyja.
- A `_set_fps(n)` később módosítja ezt a korlátozást. A 0 törli azt.

A nevek nem megkülönböztetik a kis- és nagybetűket; a szerkesztő kisbetűvel tárolja őket. A `function` és a `fun` ugyanaz a kulcsszó.

Nyelv

Programozási alapismeretek

Egy program a játék kazettájában futtatott sorok listája. Minden címsor először a sor alakját mutatja, majd egy rövid példát. A `--` kezdetű sor egy megjegyzés: egy megjegyzés számodra. A játék kihagyja.

Változók

A változó egy név, amely egy értéket tartalmaz. Te választod ki a nevet. A későbbi sorok beolvashatják, és egy új értéket tárolhatnak ugyanazon a néven.

A forma `name = value`. A `local name = value` ezt a nevet az aktuális blokkon belül tartja, így a játék többi része nem láthatja. `local` nélkül a név globális, és minden fül láthatja.

```
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
```

A `nil` azt jelenti, hogy nincs érték. A `true` és a `false` jelentése igen és nem. Egy soha nem beállított név `nil` formátumú.

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

Táblázatok

Egy táblázat sok értéket tartalmaz egy név alatt. Egy lista számokat használ. Az első elem 1, nem 0. Egy rekord neveket használ. Egy táblázat egy másik táblázatot is tartalmazhat.

Lista alakja: `{ value, value }`. Rekord alakja: `{ name = value, name = value }`. A rekordok listája egy tábla, amelynek elemei táblázatok.

```
-- 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
```

Állapot és test

A feltétel egy igen vagy nem válaszú kérdés, például `hp > 0` vagy `btnd(4)`. A törzs a sorok azon csoportja, amelyek akkor futnak le, amikor belépsz egy blokkba. A `if` és minden ciklus ugyanazt az alakzatot használja: egy feltétel, majd egy törzs, végül `end`. A `repeat until`-tel zárul a `end` helyett.

Ha

A `if` függvény csak akkor futtatja a törzsét, ha a feltétel igaz. A `elseif` egy másik feltétellel próbálkozik. A `else` függvény akkor fut le, ha minden feltétel hamis.

Alak: `if condition then`, majd a test, végül `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
```

Hurokok

Egy ciklus megismétli a törzset. A `while` először ellenőrzi a feltételt, és addig folytatja, amíg az igaz marad. A `repeat` lefuttatja a törzset, majd ellenőrzi, hogy a törzs mindig legalább egyszer lefusson. A `for` számol. Az utolsó számot tartalmazza. Harmadik szám nélkül a lépés 1, tehát ha a kezdet már a vég után van, a törzs nem fut le. A visszaszámlálás negatív lépést igényel.

```
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
```

Asztalnál sétálva

A `all` függvény a lista minden elemét meglátogatja az elsőtől kezdve. A `pairs` függvény minden egyes nevet és az adott név alatt tárolt értéket is meglátogatja.

```
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
```

Funkciók

Egy függvény egy elnevezett recept. Az alakzat `function name(arguments)`, majd egy törzs, végül `end`. A meghívást a név és a zárójelek beírásával végezzük. A `return` egy értéket küld vissza, és többet is küldhet. Az argumentumok az adott függvény lokálisai.

```
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
```

A `local function` egy olyan függvény, ami látja a körülötte lévő lokálokat. Ez egy lezárás. Egy tábla tartalmazhat ilyet, és egy ponttal hívjuk meg.

```
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
```

Igen, nem, és a választás

A `not` függvény egy feltételt vált. A `and` csak akkor igaz, ha mindkét oldal igaz. A `or` függvény akkor igaz, ha bármelyik oldal igaz. A `a and b or c` függvény a `b` feltételt választja, ha a `a` igaz, egyébként a `c` feltételt. A `cond ? a : b` függvény ugyanazt az ötletet fejezi ki egy sorban: `a`, ha a feltétel igaz, `b`, ha nem.

```
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)
```

Matek és szöveg

A `+ - * /` az összeadás, kivonás, szorzás és osztás műveletei. A `//` oszt és megtartja az egész számot. A `%` a maradék. A `^` a hatvány. A `..` két szöveget egyesít. A `#t` azt jelöli, hogy hány elem van egy listában. Az összehasonlítások a következők: `==` (azonos), `~=` vagy `!=` (különböző), és `< > <= >=`. A `x += 1` 1-et ad hozzá x-hez. Ugyanez az alakzat működik a `-= *= /= ^= %=` esetében is.

```
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
```

Ahol egy helyi lakos él

A `local` addig tart, amíg a `end` le nem zárja a blokkját. A `if` / `end` egy saját blokk. Megemlíthetsz egy lokális értéket a sor felett, amely beállítja, de amíg az a sor le nem fut, az érték `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
```

Ugrás

A `::label::` egy sort jelöl. A `goto label` oda ugrik, de csak ugyanazon a függvényen belül.

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

Röviden, az emlékezet

A `peek` és a `poke` egyetlen 16 bites címen keresztül figyeli az élő pixeleket. Nem őriznek meg a játékkazetta második másolatát. A `0x5F00`-től `0x5FFF`-ig terjedő címek regiszterek: kamera, klip (csak olvasható), rajzolási szín, kitöltési minta, a három paletta áttérképezés és az oldalszámok. A lyuk mindkét oldalán lévő 65280 bájt alkotja a pixelablakot.

Amikor a képernyő belefér ebbe a 65280 bájtba, a 64×64 és a 128×128 belefér, a 256×256 pedig nem, akkor a képernyő a 0. címen kezdődik, soronként. Ami megmarad, az a `0x5F41` által kiválasztott sprite lap (a 0 a 0. lap). Ha ez a lap nagyobb, mint a maradék hely, a `0x5F42` kiválasztja az oldalt, és a térkép nem jelenik meg az ablakban. Ha a lap belefér, a farok a `0x5F43` által kiválasztott térkép cellái, egyenként egy vagy két bájtnyi cellák, és a `0x5F44` lapozza a térképet, ha az nem fér bele a farokba.

Amikor a képernyő nem fér el, az egész ablak egyetlen 65280 pixeles oldal. A `poke(0x5F40, n)` lapoz. A `poke(0x5F41, n)`, ha az n nem 0, az n-edik munkalapot jeleníti meg, és a 0-ás munkalapnak ebben a módban nincs nézete. A `poke(0x5F43, n)`, ha az n nem 0, az n-edik térképet jeleníti meg, és ez győzedelmeskedik a munkalap felett. A 0-ás térkép ebben a módban csak `mget` és `mset`.

Egy `peek`, amely nem pixelre vagy regiszterre érkezik, 0 értéket ad vissza. Egy `poke`, amely ott van, semmit sem csinál. Sprite jelzők, SFX és DATA táblák nincsenek ebben az ablakban. Használd a `fget`, `sfxget` és `dget` kódokat.

Gyakorlati használat

Minden függvény fel van sorolva az oldal alján, a PIX37 függvények gyakorlati alkalmazásával együtt. Az egyes példák alatti linkek további információkat tartalmaznak az adott függvényekről.



Sima mozgás

Egy 128×128-as játékkazetta felülnézetből. Az 1. sprite már kirajzolva. A nyilak minden irányba gyorsulnak, a súrlódás pedig lelassít. A sprite körbetekeredik: az alját elhagyó felét felülre rajzolják, és ugyanez történik a többi éllel is.

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



A következő lépés blokkolásának ellenőrzése

Egy 64×64-es játékkazetta. A padló és a jobb oldali fal már rajta van a térképen, a 3-as sprite-tal kifestve, és a 0. jelző van beállítva ezen a sprite-on. A nyilak az 1-es sprite-ot mozgatják. A fenti szó azt mondja, hogy blokkolva van, ha a következő lépés átfedne egy tömör csempét, és szabad, ha nem.

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



Törlés átfordítással

Nyomd meg az A billentyűt. A wipe-ot a flip() rajzolja ki, soronként, a _draw() függvényen belül. A _draw függvény futása közben nem jelenik meg, így minden sor a képernyőn marad.

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



Függvénytáblázat (osztály)

Az 1. sprite már meg van rajzolva. A make_monster úgy működik, mint egy osztály. Egy táblázatot ad vissza, az update és a draw metódusok pedig megőrzik a szörny saját x és y értékét. A felül lévő szám azt mutatja, hogy milyen messzire jutott.

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



Négyféleképpen lehet leolvasni egy gombot

Az A a 4-es gomb. Tartsd lenyomva, és a töltés emelkedik. Engedd el, és a töltés visszaáll 0-ra. A fel és le mozgás ismétlődik lenyomva tartás közben, így a menüsor folyamatosan mozog. A btnd a lenyomás első képkockája, amelyet ez a játékkazetta csak a tartás kezdeteként használ.

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



Lőj az egérrel

Az 1. sprite már középen látható. Egy fehér pont a kurzor. Bal kattintás egy pixelt lő ki felé. A kerék a sarokban látható módon változtatja a sebességet.

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



Írjon be egy nevet, és erősítse meg

Nyomd meg az A gombot a mező megnyitásához. A fel és le billentyűk villogó betűket váltanak. Az A kiírja, az X törli, az Y szóköz, a B pedig megerősíti. A sor ezután kimondja a neved.

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



Kamera a lejátszón

Egy 128×128-as játékkazetta. Az 1. sprite egy csempézett padló közepén kezdődik. A nyilak mozognak. A kamera középen tartja a sprite-ot. Az x és y a sarokban nem görgethetők, mert a camera() argumentumok nélkül visszaállítja a nézetet a HUD számára.

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



Néhány sor egy vizuális regényből

A sorok DATA betűtípussal, angolul és lengyelül vannak. Az A billentyű folytatódik, vagyis a kiválasztások maradnak, amikor egy választási lehetőség látható a képernyőn. A B billentyű elhagyja a választási lehetőséget, egyébként pedig felcseréli a nyelvet. Az X billentyű megváltoztatja a nyelvet, miközben egy választási lehetőség látható. A párbeszédpanel a 8px betűtípust használja, így a lengyel betűk megjelennek.

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



Fess kandikálással

Egy 64×64-es játékkazetta, tehát a 0-s cím a bal felső képpont, a következő sor pedig a 64-gyel kezdődik. A játékkazetta pixeleket ír, sorokat másol, sorokat töröl, és kinyomtatja a visszaolvasott értékeket.

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



Egy lista, amihez hozzáadhatsz

A hozzáad egy számot, maximum 8-ig. B eltávolítja az utolsót. X eltávolítja az elsőt. A lista a képernyőn kinyomtatódik.

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



Egy szétszedett név

A játék patronja felosztja az ada,12,blue értékeket, és kinyomtatja a nevet, a kezdőbetűt (12), és a blue szót.

`cls split sub chr ord tonum tostr print _draw`



Játssz le egy hangot, majd változtasd meg

A 0. minta egy hangjegy hosszú, és minden ismétlés ugyanolyan hangerős. Az A ismétli, amikor az a hang véget ért. A B megemeli a hangmagasságot, és lejátssza azt a hangot.

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



Egy játékkazettába írt dal

Két pattern ismétlődik a 0. sorban, egy-egy az első két csatornán. A sor a 0. marad. A ch0 és a ch1 az a hang, amelyen az egyes csatornák vannak. A pattern, LEN és LOOP szerkesztésével a játék kazettájában megtekinthetjük, hogy ezek a vezérlők mit csinálnak a zene panelen.

`songget songset music musicstat cls print _init _draw`



Három szörnyeteg a DATA-ban

Az ADATOK panelen már van egy név oszlop és egy életerő oszlop is, a nyálkával, denevérrel és szemmel. Az A sebzi a nyálkát. A jobb oldali számok az életerejét jelzik.

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



Emlékezz az eredményre

A hozzáad egy pontot és elmenti. Kilép és újra kinyitja a játékkazettát: a szám még mindig ott van. B elfelejti.

`load btnd save delsave cls print _init _update _draw`




Egy második fejezet egy másik fájlban

Két játékkazetta osztozik a mentési hely küldetésén. 1. rész: Az A jelzi a játék végét. A 2. rész, amelyet először nyitnak meg, azt mondja, hogy fejezzék be az 1. részt. Ezután megnyílik a 2. rész, és a B törli a mentést, hogy az 1. rész újra játszható legyen.

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



Egy eredmény

A 8-as sprite már kihúzva van. Nyomd meg az A billentyűt tízszer. A pirítós megnevezi az elért eredményt, és a 8-as sprite marad, mert feloldották. A B billentyű törli az elért eredményt a helyi fájlból, és visszaállítja a számlálót.

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



Egy szoba, ahová be lehet lépni

Egy 256×256-os játékkazetta. Egy fehér pont a kurzor. A menü a host, a join vagy a szerverlista. A host megnyit egy walk nevű szobát és elindítja a játékot. A Join egy címet görget (számjegyek, egy pont, egy kettőspont), és B csatlakozik. A szerverlista fel és le billentyűkkel mozog, és A csatlakozik a sorhoz. A játékban a nyilak séta sprite 1. Egy érme megosztott: a host mozgatja, és ha felveszi, 1-gyel növeli a játékos pontszámát, és létrehozza a következőt. Minden sprite, beleértve a tiédet is, mellette mutatja az adott pontszámot. A sarokban látható, hogy hány játékos van bent, a LAN címe és a host nyilvános címe. B elmegy: te és a többiek visszatértek a menübe, és a következő host egy új szobát indít 0 pontszámmal.

```
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
```



Párbeszéd, ami vár

Az A a következő sort mutatja. A függvény minden egyes ütésnél szünetel, amíg újra meg nem nyomjuk az A billentyűt.

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



Várj fél másodpercet

A szó körülbelül fél másodperc múlva ...-ról szia-ra változik. A képernyő folyamatosan rajzol, amíg a korutin szünetel.

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



Körök, amelyek egyenként érkeznek

Öt kör csöpög be felülről, két képkocka távolságra egymástól, majd lehullanak.

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



Két képkocka az órából

Az 1. és 2. sprite ugyanazon karakter két képkockája. A kép másodpercenként négyszer cserélődik.

```
cls flr time spr _draw
```

Minden funkció

A szögletes zárójelben lévő argumentumok megadása opcionális. A táblázatok és karakterláncok indexei 1-gyel kezdődnek, kivéve a DATA sorokat és oszlopokat, amelyek 0-val kezdődnek. A színek palettaindexek.

Index

_rajzol
_init
_mp
_mp_fail
_mp_join
_mp_leave
_mp_update
_set_fps
_update
abs
achclear
achdef
elérni
elérte
add hozzá
minden
állítja
atan2
zenekar
bnot
bor
btn
btnd
btnp
btnu
bxor
kamera
ceil
chr
kör
körkitöltés
kattintson
clickm
klikker
klip
cls
együttteremteni
színe
törzskönyv
cos
costatus
gróf
kurzor
dcols
del
deli
delsave
dget
drow-ok
dset
dstr
fget
fillp
flip
flr
foreach
fset
info
ipairs
keyoff
keyon
keystat
keystr
vonalat
terhelést
log
lshr
térképet
max
memcpy
memset
mget
mid
min
egér
mset
zene
musicstat
bemondani
hívja fel
következő
nhost
njoin
nkb
nkbstr
nlan
nleve
nlist
nhallgass
megjegyzés
nplayers
nsend
nstun
rend
ovális
ovális kitöltés
csomag
párokat
haver
palrgb
palt
pcall
lesni
kukucskál2
kukucskál4
perf
pget
piszkálni
piszkálni2
piszkálni4
nyomtatni

nyomtatás10	setpaletta	dalkészlet
nyomtatás8	setspr	hasított
pset	settex	spr
rect	beállt	sqrt
rectfill	sfx	srand
rnd	sfxget	sset
rotl	sfxhdr	sspr
rotr	sfxset	stat
helyes	sfxstat	al
rrectfill	sget	idő
menteni	sgn	tline
mentési hely	shl	tonum
képernyő	mutatják a statisztikákat	tostr
görgetni	shr	kicsomagolni
válassza ki	bűn	whoami
setmap	dalszöveg	hozam

Életciklus

_init

`_init() ... end`

Egyszer lefut, amikor a játék kazetta elindul. A hookon kívül eső kódok is bekerülnek az _init-be.

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

_update

```
_update() ... end
```

Játéklogika. 60 Hz-en hívódik meg, még akkor is, ha a _draw korlátozott vagy a host lassabb. Olvasd el a gombokat itt.

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

_draw

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

Egy képkockát rajzol. Argumentum nélkül, vagy 0 értékkel, annyiszor fut, ahányszor a gazdagép megjeleníti. Adjon át n (1..999) értéket az n képkocka/másodperc sebességű rajzoláshoz. Az _update 60 Hz-en marad.

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

_set_fps

```
_set_fps(n)
```

Megváltoztatja a _draw korlátot játék közben. A 0 korlátlan értéket jelent. Az n 0..999-re van korlátozva.

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

flip

```
flip()
```

Megjeleníti a képernyőt, majd enged a következő képkockáig. Csak akkor használd, ha saját képkocka-ciklust rajzolsz. Ne várj utána.

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

time

```
time()
```

Másodpercek az _init óta.

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

Bemenet

btn

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

1, miközben a b gombot lenyomva tartjuk, egyébként 0. p a játékos, 0..7. Argumentumok nélkül a 0. játékos bitmaszkját adja vissza. Gombok: 0 bal, 1 jobb, 2 fel, 3 le, 4 A, 5 B, 6 X, 7 Y, 8 R, 9 L, 10 select, 11 start.

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

btnd

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

Csak az _update kereten igaz, amikor a b gomb le van nyomva.

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

btnp

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

Lenyomáskor igaz, majd lenyomva tartás közben is: 30 tikk késleltetés, majd 8 tikkenként. Gomb nélkül maszkot ad vissza.

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

btnu

`btnu(b, [p])`

Igaz, csak a kereten engedik el a gombot.

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

mouse

`mouse()`

Visszaadja az egér x és y koordinátáit a játék pixelei között.

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

clickl

`clickl([p])`

Bal egérgomb. A clickl() lenyomva tartva. A clickl(1) a lenyomási él.

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

clickr

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

Jobb egérgomb. Ugyanaz az 1 argumentumos él, mint a kattintásnál.

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

clickm

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

Középső gomb, vagy egy kattintás a görgővel.

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

scroll

```
scroll()
```

Egérgörgő mozgása ebben a keretben. A pozitív és negatív értékek a kerék irányától függenek.

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

keyon

```
keyon()
```

Megnyit egy gombokkal vezérelhető szövegmezőt. A fel és le billentyűk A–Z betűket váltanak. Az A kiírja az adott betűt. Az X törli. Az Y szóköz. A B megerősíti és bezárja a mezőt. A billentyűzettel nem lehet belegépelni.

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

keyoff

`keyoff()`

Elrejti a szövegmezőt.

```
keyoff()
```

keystr

`keystr([s])`

Szöveg a mezőben, majd 1 a kereten B megerősíti, majd a felső és alsó betűk változnak. A betű üressé válik, amint a mező bezárul. A keystr("") törli a szöveget.

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

keystat

`keystat()`

1, amíg a szövegmező nyitva van, egyébként 0.

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

Rajz

screen

`screen()`

A játék aktuális képernyőmérete pixelben. Először a szélességet, majd a magasságot adja vissza. Ez a játék felbontása, nem a szerkesztőablak felbontása.

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

cls

`cls([col])`

Törli a képernyőt. Az alapértelmezett szín a 0.

```
cls(1)
```

color

`color(col)`

Beállítja a későbbi hívásokban használt rajzolási színt, ha a col paramétert kihagyjuk. Argumentum nélkül az aktuális színt adja vissza.

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

camera

`camera([x, y])`

Minden draw hívást eltol. A camera() visszaállítja 0, 0 értékre.

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

clip

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

A rajzolást egy téglalapra korlátozza. Egy nullától eltérő ötödik argumentum metszi az aktuális klipet. A clip() visszaállítja a teljes képernyőt. A klipregiszterek betekintése csak olvasható.

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

cursor

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

Beállítja, hogy hová kerüljön a következő koordináták nélküli nyomtatás, és opcionálisan a színt is. A cursor() visszaadja az x és y értékeket.

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

pset

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

Egy képernyőképpontot állít be. Beleszámít a px statisztikába.

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

pget

`pget(x, y)`

A képernyő pixelének színe x és y koordinátáknál.

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

print

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

3×5-ös szöveget rajzol (UTF-8, a Fusion betűtípusból kivágva). X és y nélkül a kurzornál nyomtat, majd a következő sorba ugrik.

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

log

`log(...)`

A szóközökkel elválasztott argumentumokat egy szövegpufferhez fűzi hozzá. A pix37_run kiírja a puffert a standard kimenetre. Nem rajzol.

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

print8

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

Ugyanaz, mint a nyomtatás, csak a 8 px-es Fusion betűtípussal.

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

print10

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

Ugyanaz, mint a nyomtatás, csak a 10 px-es Fusion betűtípussal.

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

line

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

Egyenes vonal x0,y0-tól x1,y1-ig. Két argumentum folytatódik az előző végétől. A line() visszaállítja ezt a láncot.

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

circ

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

Kör körvonala. A sugár képpontban van megadva.

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

circfill

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

Kitöltött kör.

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

oval

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

Ellipszis körvonal a határoló dobozon belül. A sarkok beleértendők.

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

ovalfill

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

Kitöltött ellipszis egy határoló dobozban.

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

rect

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

Téglalap alakú körvonal. A sarkok beleértendők.

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

rectfill

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

Kitöltött téglalap.

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

rrect

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

Lekerekített téglalap körvonala. x és y a bal felső sarok. w és h a méret pixelben; mindkettőnek nagyobbnak kell lennie, mint 0. r a sarok sugara, alapértelmezett értéke 0. A rövidebb oldal felére van rögzítve.

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

rrectfill

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

Kitöltött, lekerekített téglalap. Ugyanazok az argumentumok, mint a „helyes” esetén.

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

fillp

`fillp([p])`

16 bites kitöltési minta később kitöltött alakzatokhoz. A 0 tömör értéket jelent. A fillp() az aktuális mintát adja vissza. A bitek a 4×4-es mintát követik.

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

pal

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

A c0 színt c1-re képezi át. A 0-s mód a rajzoló paletta, az 1-es a megjelenítési paletta, a 2-es pedig a kitöltési minta paletta. A pal() visszaállítja az értékeket.

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

palrgb

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

Egy palettabejegyzést RGB-re állít, minden csatornát 0..255-re.

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

palt

`palt([col, t])`

Átlátszóság. A palt(col) vagy a palt(col, true) átlátszóvá teszi a col-t. A palt(col, false) tömör színt ad neki. A palt() minden színt átlátszatlanná tesz.

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

setpalette

`setpalette(n)`

Kiválasztja az n palettabankot. A setpalette() visszaadja az aktuális bankot.

```
setpalette(1)
```

tline

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

Vonalat rajzol a képernyőn, miközben a program a térképpontokat mintavételezi az mx, my, mdx és mdy térképpozíciókból, és a térképmintát a vonal mentén lépteti.

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

Sprite-ok és térképek

spr

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

Az n sprite-ot kirajzolja x és y koordinátákon. A w és h a méret pixelben, alapértelmezett érték 8. Az fx és fy vízszintesen és függőlegesen tükrözik, ha nem nulla. A tex a sprite-lap tabulátora. A rot a szög fordulatokban (0..1), ugyanabba az irányba, mint a sin és cos. A 0 érték a sprite-ot nem forgatja el.

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

sspr

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

A sprite lap egy téglalapját másolja a képernyőre, dw és dh méretarányban. Az alapértelmezések megtartják a forrásméretet.

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

sget

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

Egy texel színe a sprite lapon.

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

sset

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

Egy texelt ír a sprite lapra.

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

fget

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

Sprite jelzők. Az fget(n) visszaadja a jelző bájtot. Az fget(n, f) igaz, ha az f (0..7) jelzőbit be van állítva.

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

fset

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

Beállítja a sprite flageket. Az fset(n, v) kiírja a teljes bájtot. Az fset(n, f, v) egy bitet állít be.

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

setspr

```
setspr(n)
```

Kiválasztja az aktív sprite lapot. A setspr() visszaadja az aktuális lapot. Ugyanaz a bank, mint a settex-szel.

```
setspr(1)
```

settex

```
settex(n)
```

A setspr álneve.

```
settex(0)
```

map

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

Kirajzolja a térképet. A térkép cx, cy cellái az sx, sy képernyőn kerülnek kirajzolásra. A cw, ch értékek a 0-t jelentik, ami a teljes térképet jelenti. A réteget elfogadja és figyelmen kívül hagyja. A tex a csempekészlet lapja.

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

mget

`mget(cx, cy)`

Sprite azonosító a cx és cy térképcellában.

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

mset

`mset(cx, cy, n)`

Az n sprite-ot beírja a cx, cy térképcellába.

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

setmap

`setmap(n)`

Kiválasztja az n. térképet. A setmap() visszaadja az aktuális térképet. A 0. térkép egy valós térkép.

```
setmap(1)
```

Memória

peek

`peek(addr)`

Beolvas egy bájtot a 64 KB-os ablakból. Az ablak az élő képernyőt tartalmazza, majd sprite-képpontokat készít, majd amikor a képernyő elfér 65280 bájtban, cellákat képez le. A nagyobb képernyőket a poke(0x5F40, page) paranccsal lapozhatja. A nem leképezett címek 0-t adnak vissza. @addr néven is írható.

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

poke

`poke(addr, ...)`

Bájtokat ír a címre. A további argumentumok a következő bájtokat írják. A nem leképezett címeket figyelmen kívül hagyja. Szintén kiírja a $addr, val értéket. A képernyő bökése pixelírásnak számít.

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

peek2

`peek2(addr)`

Előjel nélküli 16 bites érték, kis végű, a címen.

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

poke2

`poke2(addr, val)`

Egy előjel nélküli 16 bites értéket ír ki, kis-endiánnal.

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

peek4

`peek4(addr)`

Előjel nélküli 32 bites érték, kis végű.

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

poke4

`poke4(addr, val)`

Egy előjel nélküli 32 bites értéket ír ki, kis-endiánnal.

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

memcpy

`memcpy(dst, src, len)`

Len bájtokat másol ugyanabba a 64 KB-os ablakba.

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

memset

`memset(dest, val, len)`

A len bájtokat a céltól kezdődően val értékkel tölti fel.

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

Matek

abs

`abs(x)`

Abszolút érték.

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

flr

`flr(x)`

Mínusz végtelen felé kerekít.

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

ceil

`ceil(x)`

Plusz végtelen felé kerekít.

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

max

`max(a, b)`

A két érték közül a nagyobb.

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

min

`min(a, b)`

A két érték közül a kisebb.

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

mid

`mid(a, b, c)`

A három érték közül a középső. Szorítóként hasznos.

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

sgn

`sgn(x)`

Az x jele: -1, 0 vagy 1.

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

sqrt

`sqrt(x)`

Négyzetgyök. A negatív argumentum nem szám.

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

sin

`sin(x)`

A szinusz. Az x felváltva van: 0, 0,25, 0,5, 0,75 egyszer körbejárja a kört. Nem radiánban.

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

cos

`cos(x)`

A koszinusz viszont ugyanazon konvenció szerint értelmezendő, mint a szinusz.

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

atan2

```
atan2(dx, dy)
```

A dx és dy vektorok szöge 0-tól 1-ig terjedő fordulatokban.

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

rnd

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

Véletlenszerű szám a [0, x] intervallumban. Az alapértelmezett x 1, tehát az rnd() értéke 0..1. Kocka esetén flr(rnd(6))+1.

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

srand

```
srand(seed)
```

Beindítja a véletlenszám-generátort, hogy egy későbbi mólsorozat megismételhető legyen.

```
srand(1)
```

band

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

Bitenkénti ÉS. Írásban a és b is szerepel.

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

bor

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

Bitenkénti VAGY. Más néven a | b.

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

bxor

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

Bitenkénti XOR. Más néven ^^ b.

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

bnot

```
bnot(a)
```

Bitenként NEM. Írásban ~a.

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

shl

```
shl(a, n)
```

Balra tolódás. Írásban is szerepel << n.

shr

```
shr(a, n)
```

Számtani eltolás jobbra. Más néven >> n.

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

lshr

`lshr(a, n)`

Logikai jobbra eltolás. >>> n-ként is írják.

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

rotl

`rotl(a, n)`

Biteket forgat balra. Írják még <<> n-ként is.

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

rotr

`rotr(a, n)`

Forgasd el a biteket jobbra. Más néven >>< n.

Táblázatok és értékek

add

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

Hozzáfűzi a v-t a t táblához. Ha az i adott, akkor az adott 1-alapú indexnél szúrja be.

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

del

`del(t, v)`

Eltávolítja az első, v-vel egyenlő bejegyzést.

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

deli

```
deli(t, [i])
```

Eltávolítja az i indexet és visszaadja annak értékét. Ha nincs i, akkor eltávolítja az utolsó bejegyzést.

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

all

```
all(t)
```

Iterátor a tömb részén, sorrendben. A lyukak megállítják a sétát, ahogy egy 1-alapú tömb véget ér.

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

foreach

```
foreach(t, fn)
```

Minden tömbelemre meghívja az fn(v) függvényt.

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

pairs

```
pairs(t)
```

Iterátor minden kulcson és értéken, beleértve az elnevezett mezőket is.

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

ipairs

```
ipairs(t)
```

Iterátor a tömbrészen az 1-alapú indexszel.

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

next

```
next(t, [k])
```

A k utáni következő kulcsot és értéket adja vissza. A next(t) az elejétől kezdődik. Párok használják.

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

count

```
count(t, [v])
```

A tömbrész hossza. Ha v értéke adott, akkor megszámolja, hogy hány elem egyenlő v-vel.

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

pack

```
pack(...)
```

Becsomagolja az argumentumokat egy új táblázatba.

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

unpack

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

A t[i]-től t[j]-ig terjedő tartományokat külön értékként adja vissza. Az alapértelmezett értékek a teljes tömbrészt lefedik.

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

select

```
select(i, ...)
```

A select("#", ...) visszaadja, hogy hány argumentum következik. A select(i, ...) visszaadja az i-től kezdődő argumentumokat.

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

assert

```
assert(c, [msg])
```

Hibával leáll, ha a c értéke hamis vagy nil. Az msg lecseréli az alapértelmezett szöveget.

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

pcall

```
pcall(f, ...)
```

Meghívja az f függvényt. Siker esetén igaz értéket és az eredményt adja vissza. Hiba esetén hamis értéket és a hibaüzenet szövegét adja vissza.

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

Húrok

chr

`chr(n)`

Egy karakter a 0..255 bájtkódból.

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

ord

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

Az i-től j-ig terjedő indexű s bájtkódok. Az indexek 1-alapúak. Alapértelmezés szerint az első karakter.

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

sub

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

i-től j-ig terjedő részkarakterlánc, 1-alapú. Hagyja ki a j-t, hogy a vége legyen.

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

split

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

Felosztja az s függvényt egy táblázatba. A sep alapértelmezett értéke vessző. A sep egy szám, ami egy adathalmaz szélességét jelöli. A conv tonum értékkel konvertálja a darabokat, ha igaz.

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

tonum

`tonum(s, [base])`

Egy számot elemez. Az alapérték 2–36 lehet. Null értéket ad vissza, ha a szöveg nem szám.

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

tostr

`tostr(v, [digits])`

A *digits* (digits) szöveggé alakítja az értéket. A *digits* (digits) a számok tizedesjegyeit állítja be.

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

Hang

sfx

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

Lejátssza az n mintát és visszaadja a csatornát. Ha a ch karakterláncot elhagyjuk, vagy átadjuk a -1 paramétert, akkor az első szabad csatorna lesz kiválasztva. A ch -2 leállítja az n mintát minden csatornán. Az n -1 leállít egy csatornát, vagy minden csatornát, ha a ch karakterlánc kimarad. Az n -2 lehetővé teszi a ciklusos minta befejezését a ciklus helyett. Az argumentumok nélküli sfx() -1 értéket ad vissza.

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

sfxget

`sfxget(n, step)`

Egyetlen pattern lépésből álló táblázatot ad vissza: [1] hangmagasság, [2] hullámforma, [3] hangerő, [4] effekt. A hangmagasság 0..63 (C-0-tól D#5-ig). A hullámforma 0..15. A hangerő 0..7, a 0 pedig néma. Az effekt 0..7.

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

sfxset

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

Egy mintalépést ír. Mind a hat argumentum megadása kötelező.

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

sfxhdr

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

Minta fejléc. Az sfxhdr(n) visszaadja a fejléc táblázatot. Az sfxhdr(n, key, val) egy mezőt állít be: speed, loop_in, loop_out, level, noise, crush, dtune, rverb, dampn, swing, steps. A 0..10 közötti számok ugyanazokat a mezőket választják ki ebben a sorrendben. A speed a TIC, azaz lépésenkénti tick. A magasabb érték lassabb. A 15 120 ütést jelent percenként. A 16 bites 1..255 közötti értékeket fogad el. A szűkebb szintetizátorok a TIC listájukhoz igazodnak.

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

sfxstat

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

A csatornán lejátszott minta azonosítója: ch, vagy -1. Az alapértelmezett csatorna a 0.

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

note

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

Egy hangot játszik le minta nélkül. hangmagasság 0..63, hullámforma 0..15, hangerő 0..7, effekt 0..7. A tiktek száma adja meg, mennyi ideig tart, ugyanolyan 120 Hz-es tiktekben, mint a TIC. Ha kihagyod a tiktek számát, akkor 15 percig tart, egy alapértelmezett lépésben. Az effekt ennyi ideig fut, majd a csatorna elhallgat.

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

music

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

Elindítja az n. dalt felülről. A music(-1) leáll. A már lejátszott azonosítójú music meghívása futtatást eredményez. A music() visszaadja a lejátszott dal azonosítóját, vagy -1-et. A fade időtartama ezredmásodperc: a dal akkor halkul el, vagy halkul el, amikor n értéke -1. A beállított maszk bitek a dalhoz használt csatornák. A 0 azt jelenti, hogy mind a nyolc. A többi csatorna szabadon marad az sfx számára.

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

musicstat

```
musicstat()
```

A lejátszott dal táblázata. A „song” és a „row” a dal azonosítója és a sor, vagy -1, ha semmi sem szól. A „step” és a „pos” nyolc számjegyből áll. A „step[1]” a 0-as csatorna, a „step[8]” a 7-es csatorna. A „step” a hangjegy indexe, vagy -1, ha a csatorna néma. A „pos” azt jelzi, hogy a hang milyen messzire jutott el 0-tól 1-ig. A „tick” azt jelzi, hogy ez a sor hány tick-et játszott már le. A „played” azt jelzi, hogy hány sor kezdődött el a music() függvény futtatása óta, beleértve az éppen lejátszott sort is.

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

songget

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

A minta azonosítója egy dalcellában tárolódik. A ch értéke 0..7. A 255 azt jelenti, hogy a cella üres. Ez az élő hívás; a kódban található tipp szövege továbbra is egy régebbi argumentumsorrendet mutat.

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

songset

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

Egy mintaazonosítót ír egy dalcellába. A cella törléséhez használd a 255-ös kódot.

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

stat

`stat(n)`

Csak hang. A 16..23 az n-16-os csatorna sfx azonosítója. A 24..31 az n-24-es csatorna zenei mintája. Bármely más n 0-t ad vissza. A képkockaidőzítés és a RAM perf és info, nem stat.

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

Adattáblák

dget

`dget(t, row, col)`

Beolvassa a t adattábla egy celláját az ADATOK panelről. A sor és az oszlop 0-val kezdődik.

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

dset

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

Cella kiírása. Szöveges oszlopként karakterlánc használata.

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

dstr

`dstr(t, row, col)`

A cella szövegként jelenik meg. A számok konvertálva vannak.

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

drows

`drows(t)`

Hány sora van a t táblának?

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

dcols

`dcols(t)`

Hány oszlopa van a t táblának?

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

Mentés és eredmények

save

`save(k, v)`

A v értéket a k kulcs alatt tárolja a mentési fájlban. A v lehet szám, karakterlánc vagy ezek táblázata. A fájl tömörített.

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

load

`load(k)`

Beolvassa a k kulcsot. nil értéket ad vissza, ha a kulcs hiányzik.

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

delsave

`delsave(k)`

Törli a k kulcsot a mentésből.

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

saveslot

`saveslot([name])`

Kiválasztja a mentési fájlt a szerző mappájában. Név nélkül a hely a játék címe. 1-et ad vissza, ha a fájl már létezik.

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

achieve

`achieve(i)`

Feloldja az i. achievementet 0-tól 15-ig. Helyileg tárolva. A Steam build az ACH_i-t is beállítja. Egy katalógus munkamenet jelenti a játékos fiókjának.

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

achieved

```
achieved(i)
```

1, amikor az i. eredmény már feloldva van.

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

achdef

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

Pohárköszöntő szöveg az i. teljesítményhez, 15 másodpercig látható a képernyő aljától számítva. Az spr egy opcionális 8×8-as csempe, tex a lapja.

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

achclear

```
achclear(i)
```

Törli az i. achievementet a helyi fájlban. A már beállított Steam ACH_i beállítva marad.

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

Hálózat

whoami

```
whoami()
```

A játékosazonosítód 0..15, amíg a játék folyamatban van. Offline -1.

```
id = whoami()
```

nhost

```
nhost()
```

1, ha ez a gép a gazdagép, egyébként 0.

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

nplayers

```
nplayers()
```

Hány élő azonosító létezik a 0..15 tartományban?

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

nlisten

```
nlisten()
```

Elindítja az UDP hosztolását és ip:port formátumban adja vissza a LAN címet.

```
addr = nlisten()
```

njoin

```
njoin(addr)
```

Csatlakozik az ip vagy ip:port címhez.

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

nleave

```
nleave()
```

Kilép a munkamenetből. A partnerek a _mp_leave üzenetet a 0-s miérttel látják.

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

nsend

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

Veszteséges üzenetet küld. A név maga az esemény. Legfeljebb négy további érték követheti. p 0-tól 15-ig suttog az adott partnernek; p elhagyása esetén mindenkinek elküldi.

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

ncall

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

Ugyanolyan alakú, mint az nsend, de az üzenetet újrapróbálja a rendszer, amíg vissza nem nyugtázzák. Három egymást követő hiba megszakítja a munkamenetet.

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

nlan

`nlan()`

Aktuális LAN ip:port az nlisten után.

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

nstun

`nstun()`

A STUN-ból származó nyilvános ip:portod. Egy másik hálózaton lévő vendég megpróbálhat csatlakozni hozzá.

```
pub = nstun()
```

setturn

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

TURN tartalék függvény, ha egy közvetlen elérési út nem működik. Az üres argumentumokkal rendelkező setturn() kikapcsolja. A cím: host:port.

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

nlist

`nlist(game)`

A játéknévhez tartozó nyitott munkamenetek táblázata. Minden bejegyzés tartalmazza a címet, a játékosokat és a nevet.

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

nannounce

`nannounce(game)`

Szívverés, hogy ez a host megjelenjen a játék nlistájában. Hívd meg időzítőn hostolás közben.

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

nkb

`nkb([on])`

Cím mező. Az nkb(1) megnyitja, az nkb(0) elrejti, az nkb() pedig megmutatja, hogy meg van-e nyitva. A fel és le billentyűk karaktereket váltanak, 0–9, pontot vagy kettőspontot. Az A kiírja. Az X törli. A billentyűzet nem gépel bele.

```
nkb(1)
```

nkbstr

`nkbstr([s])`

A cím mezőben lévő szöveg, majd a felfelé és lefelé mutató karakterek változnak. Az nkbstr("") törli a szöveget.

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

_mp

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

Ezt a függvényt te írod. Akkor fut le, amikor egy üzenet megérkezik. A „from” a küldő azonosítója, a „name” pedig az esemény. Legfeljebb négy érték követheti.

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

_mp_join

`_mp_join(p)`

Te írod ezt a függvényt. Egy peer p azonosítót kapott. Ez neked is aktiválódik; a whoami() az az azonosító.

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

_mp_leave

```
_mp_leave(p, why)
```

Ezt a függvényt írod. A p partner balra megy. Miért a 0 egy normál kilépést jelent. Miért az 1 egy megszakadt kapcsolatot?

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

_mp_update

```
_mp_update()
```

Te írod ezt a függvényt. 15 _update keretenként lefut, körülbelül 4 Hz-en. Jó hely a pozíciók nsend küldésére.

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

_mp_fail

```
_mp_fail()
```

Ezt a függvényt írod. A munkamenet megszakadt: a gazdagép eltűnt, vagy az ncall háromszor is meghiúsult.

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

Korutinok

cocreate

```
cocreate(f)
```

Létrehoz egy korutint az f függvényből. Nem fut le, amíg a coresume parancsot nem kapjuk meg.

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

coresume

`coresume(c, ...)`

A c korutint addig futtatja, amíg eredményt nem ad, vagy vissza nem tér. Siker esetén: igaz és a kapott értékek. Hiba esetén: hamis és a hiba.

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

costatus

`costatus(c)`

„felfüggesztve”, „fut” vagy „halott”.

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

yield

`yield(...)`

Visszaadja az irányítást annak, aki a coresume függvényt meghívta. Az értékek a coresume függvény eredményei lesznek.

```
yield(1)
```

Intézkedés

info

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

Kirajzolja az élő HUD-ot a _draw-ban: fps (1 másodperces átlag), upd, drw, px, max, ram, peak, ramc és net, amikor egy munkamenet véget ért. Az info() a nyomtatási kurzort használja. Az info(col) csak a színt állítja be. Az info(x, y) vagy az info(x, y, col) elhelyezi. A számok B, k vagy m betűket használnak, hogy elférjenek egy kis képernyőn. Maga a hívás nem számolódik px-ben.

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

showstats

`showstats([on])`

Ugyanaz a HUD, mint az info-nál, automatikusan kirajzolódik 0-nál, 0 a _draw után, figyelmen kívül hagyva a kamerát és a klipet. A showstats(0) elrejti. A showstats(1) vagy a showstats() megjeleníti.

```
showstats(1)
```

perf

`perf(n)`

Egy HUD szám, hogy magad is megrajzolhasd. 0 fps, 1 upd, 2 draw, 3 px, 4 max (a legnehezebb képkocka ebben a másodpercben), 5 RAM, 6 RAMC (RAM plusz a becsomagolt játékkártya), 7 peak RAM a játékkártya indulása óta.

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