# Priročnik za PIX37

Najprej urejevalnik, nato programiranje. Neposredna kopija knjige na pix37.party/docs.

# Urednik

PIX37 DEV je urejevalnik. Tukaj rišete, pišete in poslušate igro, nato pa izvozite igralni vložek ali samostojno gradnjo. Prav tako lahko predvaja igralni vložek in odpre PIX37.PARTY za prenos igralnih vložkov in uporabo kataloga. PIX37 DEV EDUCATIONAL je isti urejevalnik in je brezplačen. Nima gradenj, izvoza igralnih vložkov in dostopa do PIX37.PARTY. PIX37 Console je predvajalnik. Predvaja igralne vložke `.p37`, vi pa odprete PIX37.PARTY za prenos igralnih vložkov in brskanje po katalogu.

## Projekti in igralne kartuše

Datoteka `.pix37` je projekt: izvorni zavihki, listi spritov, zemljevidi, zvok in podatkovne tabele. Datoteka `.p37` je igralni vložek, ki ga igralec izvaja: tesno zapakirana bajtna koda.

Shranjene datoteke in dosežki se nahajajo v mapi avtorja poleg podatkov aplikacije. V sistemu Windows je to videti kot `C:\\Users\\Ada\\AppData\\Roaming\\PIX37\\Mira\\Pong.p37s` za shranjevanje in `C:\\Users\\Ada\\AppData\\Roaming\\PIX37\\Mira\\Pong.ach` za dosežke. V Linuxu sta isti datoteki `~/.local/share/PIX37/Mira/Pong.p37s` in `~/.local/share/PIX37/Mira/Pong.ach`. Ada je račun v računalniku. Mira je ime avtorja in to ime je mapa. Pong je reža za shranjevanje, ki je privzeto nastavljena na naslov igre. `save` zapiše `.p37s`. Prvi odklenjeni dosežek zapiše `.ach`. Če so datoteke že tam, jih tekač prebere, ko se igralna kartuša zažene.

## Razvijalski meni

Tipka Escape odpre meni za razvoj. Tipka Escape med delovanjem igre vrne v urejevalnik. Potrdite z Enter ali tipko A (privzeto V). Tipka Back je tipka B (privzeto C).

Če projekt ni naložen, je meni naslednji: USTVARJANJE PROJEKTA, NALAŽENJE PROJEKTA, IZGRADNJA, NALAŽENJE igralne kartuše, POVEŽI SE S PIX37.PARTY, NASTAVITVE.

Ko je projekt naložen, sledijo naslednje koraki: SHRANI PROJEKT, NALOŽI PROJEKT, USTVARI PROJEKT, SREDSTVA, GRADI, NALOŽI igralno kartušo, POVEŽI SE S PIX37.PARTY, NASTAVITVE.

Ctrl+S shrani odprt projekt. Ctrl+R ga zažene.

## Ustvari projekt

CREATE PROJECT zahteva igralno kartušo, preden začnete ustvarjati projekt.

- **VIDEO** — 1-BITNI (2 barvi), 2-BITNI (4), 4-BITNI (16), 8-BITNI (256).
- **ZVOK** — globina vzorca sintetizatorja: 16, 8, 4, 2 ali 1 bit.
- **ZASLON** – prednastavitve vključujejo 64 × 64, 128 × 128, 160 × 90, 256 × 256 in 320 × 180. Velikost PO MERI vam omogoča izbiro poljubne velikosti od 1 × 1 do 512 × 512.
- **PALETE** – nabori barv. Prvi nabor se začne kot spodnji nabor 16 barv. NALOŽI PALETO uporabi vgrajeni nabor: PICO-8 16, PICO-8 32, Game Boy, 1-bit, CGA, Sweetie-16 in daljši seznam, vzet iz Lospeca (Resurrect 64, Apollo, Endesga 32 in 64, CC-29, SLSO8, Pear36, Oil 6, Duel, AAP-64, Lost Century in drugi). Lahko naložite tudi datoteko palete ali pa naložite svoj PNG in vzamete paleto iz te slike.




Barvni indeksi, ki jih uporabljajo `cls`, `pset`, `spr` in ostali. Kateri koli indeks lahko prebarvate v urejevalniku palet.

## Pet panelov

Urejevalnik je sestavljen iz GAME (GAME), PAINT (SLIKAR), CODE (KODA), SOUND (ZVOK) in DATA (PODATKI), v tem vrstnem redu. Zemljevid se nahaja znotraj PAINT (SLIKAR). SFX (ZVOK) in pesmi se nahajajo znotraj SOUND (ZVOK). Kliknite ploščo, da tam delate. Ko je v fokusu CODE (KODA), SOUND (ZVOK) in DATA (PODATKI) prezreta črkovne tipke, zato tipkanje ne igra hkrati tudi klavirja.



> Polna postavitev, zapisana kot `1:1:4:1:1`. Vsaka številka predstavlja, koliko vrstice zavzema ta plošča: GAME, PAINT, CODE, SOUND, PODATKI. Majhen zaslon je na sredini celice velikosti 128 × 128. Zaslon, večji od 128 slikovnih pik, razširi stolpec GAME.

NASTAVITVE → GLOBALNO → POSTAVITEV spreminja razporeditev:

| Koda | Kar vidiš |
|---|---|
| `1:1:4:1:1` | Vse. IGRA in ZVOK naložena drug na drugega, BARVA in PODATKI naloženi drug na drugega, KODA na desni. |
| `1:1:4:1:1 2X` | Ista razporeditev. KODA je narisana z dvakratno gostoto, zato se vanjo prilega dvakrat več besedila. |
| `1:0:1:0:1` | IGRA, PODATKI, KODA. |
| `1:0:1:0:1 2X` | Ista ureditev, z gostejšo KODO. |
| `0:1:1:1:1` | Brez predogleda igre. BARVA, KODA, ZVOK, PODATKI. |
| `0:1:1:1:1 2X` | Ista ureditev, z gostejšo KODO. |
| `0:1:4:1:2` | Studio. Širok KODA, višji PODATKOVNI trak, BARVA in ZVOK. |
| `0:1:4:1:2 2X` | Ista ureditev, z gostejšo KODO. |
| `0:0:4:0:4` | Samo KODA in PODATKI, oba velika. |
| `0:0:4:0:4 2X` | Ista ureditev, z gostejšo KODO. |
| `0:1:4:1:0` | BARVA, ZVOK in široka KODA. |
| `0:1:4:1:0 2X` | Ista ureditev, z gostejšo KODO. |
| `0:4:0:0:0` | Samo BARVA, 256×256, en slikovni pik lista na en slikovni pik platna. |

Vrstica `2X` ohrani iste plošče. Plošča CODE je narisana z dvakratno ločljivostjo in pomanjšana nazaj v isto celico, tako da so besedilo, zavihki in rob roba polovično visoki in na zaslonu se prilega dvakrat toliko izvorne kode.

## IGRA

Predogled delujoče igralne kartuše. Kliknite ploščo GAME ali pritisnite Ctrl+R, da zaženete predogled. Ploščo konfigurirate v NASTAVITVE → PLOŠČA GAME:

- **Predogled** – barva v stanju mirovanja, neprekinjeno ali samodejno ponovno nalaganje med urejanjem.
- **Igraj v novem oknu** – igra se odpre poleg urejevalnika.
- **Prikaz** – predogled kot RGB888 ali RGB565.
- **IGRA 64** – zaslon velikosti 64 × 64 je bodisi centriran v razmerju 1:1 v celici bodisi pomanjšan 2×, da jo zapolni.
- **Senčilniki** — CRT in filtri v igri, ločeni od filtrov v urejevalniku.
- **Barva** — ozadje plošče.

## BARVA

Listi spritov in na strani zemljevida iste plošče še ploščice zemljevidov. Listi so lahko večji od zaslona. Velikosti predogleda spritov so 8, 16, 32, 64 in 128 (128 samo, če je list velik vsaj 128 slikovnih pik). Za povečavo pomaknite kolesce miške čez predogled spritov. Velikosti čopičev so 1, 2, 3 in 4.

| Ključ | Orodje |
|---|---|
| B | Svinčnik |
| M | Izbira pravokotnika |
| V | Laso |
| S | Žig |
| G | Vedro |
| Jaz | Kapalka |
| V | Premakni |
| L / Shift+L | Črta / krivulja |
| U / Shift+U | Votli pravokotnik / votla elipsa |
| F / Shift+F | Zapolnjen pravokotnik / zapolnjena elipsa |
| D | Tresenje |
| [ ] | Manjša/večja krtača |
| Ctrl+Z / Ctrl+Y | Razveljavi / ponovi. Ctrl+Shift+Z prav tako ponovi. |
| Ctrl+C / Ctrl+X / Ctrl+V | Kopiraj, izreži, prilepi |
| Ctrl+A | Izberite platno |

Zemljevid uporablja isto idejo, z manj orodji: B svinčnik, M pravokotnik izbira, S žig, V premikanje in Ctrl+Z, Ctrl+Y, Ctrl+C, Ctrl+V. Držite tipko Alt za povečavo teksture zemljevida. Velikost čopiča na zemljevidu ostane enaka kot pri orodjih za slikanje.

Vsak sprite ima osem zastavičnih bitov. Paint jih nastavi na sprite; `fget` in `fset` ju prebereta v igri.

NASTAVITVE → UREJEVALNIK BARVANJA je mesto, kjer se konfigurira ta plošča:

- **Paleta** — katera banka je narisana, ter nalaganje in shranjevanje.
- **Predogled** — RGB888 ali RGB565.
- **Kurzor čopiča** – kazalec med slikanjem.
- **Pripenjanje slikovnih pik** – kako se čopič pripne na slikovne pike.
- **Velik čopič** – velik čopič se zaskoči v zgornjem levem kotu ali s celotnega odtisa.
- **Čebulna lupina** - duh drugih okvirjev in v katero smer je ta duh obrnjen.
- **Povečava zemljevida** – pravi slikovni elementi ali brezplačna povečava.
- **Barve** — barve uporabniškega vmesnika za slikanje.

## KODA

Zavihki so ločene datoteke znotraj projekta. Kliknite ime zavihka, da ga preimenujete; Enter potrdi. Več zavihkov, ki vsak definira `_init`, `_update` ali `_draw`, je združenih v vrstnem redu zavihkov: prvi zavihek se izvede prvi, zadnji zavihek pa zadnji. Koda, ki ni znotraj kljuke, postane del `_init`. Igralna kartuša brez kljuke se obravnava kot ena `_draw`.

| Ključ | Dejanje |
|---|---|
| Puščice | Premakni kazalko. S tipko Shift razširiš izbor. |
| Ctrl+levo / desno | Z besedo. |
| Domov / Konec,<br>Ctrl+Domov / Konec | Začetek ali konec vrstice oziroma tabulatorja. |
| Zavihek | Vstavite dva presledka. Na seznamu za dokončanje sprejmite ime. |
| Vnesite | Nova vrstica, zamaknjena kot prejšnja. Sprejema dopolnitev. |
| Ctrl+C / X / V | Kopiraj, izreži, prilepi. |
| Ctrl+Z / Ctrl+Y | Razveljavi / ponovi. |
| Ctrl+A | Izberite zavihek. |
| Ctrl+D | Podvojite črto ali izbor navzdol. |
| Ctrl+/ | Komentirajte ali odkomentirajte s `--`. |

| | |
|---|---|
| Ctrl+F | Najdi. F3 skoči na naslednji zadetek. Escape zapre vrstico. |
| Ctrl+G | Pojdi na številko vrstice. |
| Pobeg | Najprej zapre iskanje, pojdi na ali seznam namigov. Drugi Escape zapusti urejevalnik in odpre meni za razvijalce. |

Med vnašanjem imena funkcije se prikažejo namigi. S tipko Enter ali Tab vstavite ime, s tipko `(` pa je v nastavitvah vklopljena možnost COMPLETE (. NASTAVITVE → UREJEVALNIK KODE:

- **Iskanje** – Ctrl+F v aktivnem zavihku ali na vsakem zavihku.
- **Visoko besedilo** – obrežite glife na 5 slikovnih pik ali prikažite celoten glif.
- **Velikost niza** — velikost pisave nizovnih literalov.
- **Pisava komentarjev** – 3×5 ali pisava Fusion trenutnega jezika uporabniškega vmesnika.
- **Dokončaj (** — ali dopolnjevanje vnese oklepaj.
- **Namigi za kodo** – vklopljeni ali izklopljeni.
- **Jezik namigov** – jezik menija ali angleščina.
- **Presledek streške** — dodaten odmik v praznih vrsticah.
- **Žleb** – številke vrstic.
- **Barva** — ozadje, komentarji, tabulatorji, številke, operatorji, ključne besede, nizi, funkcije motorja, vaše funkcije, spremenljivke, oklepaji, kazalka, izbor, vrstica z napako, opozorilo.

## ZVOK

Plošča ima dva zavihka. SFX je en vzorec: kratek niz not. SOUND je pesem: te vzorce postavi na osem kanalov. TIC je, koliko tikov čaka en korak. Takt je 120 tikov na sekundo, en korak pa je šestnajstinska nota, zato je 15 tikov 120 utripov na minuto. Višji TIC je počasnejši vzorec. Dva vzorca ostaneta v koraku, ko en TIC deli drugega: 15 in 30 delita drugega, 15 in 8 pa ne.

Klavir igra, ko z miško izberete noto. Bele tipke so ZXCVBNM (CDEFGAB). Črne tipke so SDGHJ (C# D# F# G# A#). Premaknite miško nad katero koli vrednost in vnesite številko ali pa jo premaknite in pomaknite kolesce. Številke višin tona v kodi segajo od 0 (C-0) do 63 (D#5).

V enem vzorcu sta LOOP in LEN isti številki. OUT nad 0 se zanka od IN do OUT, naslov pa se glasi LOOP. OUT pri 0 in IN nad 0 je LEN: toliko not se zaigra od začetka, nato se vzorec konča, naslov pa se glasi LEN. Oba pri 0 zaigrata vsak korak.

Koliko sintetizatorja lahko urejate, je odvisno od bitne globine, izbrane za kartušo igre. NASTAVITVE → UREJEVALNIK SFX nastavi to globino, ali se nota samodejno premakne naprej (izklopljeno, veriga not ali veriga polj), ali preslednica ponovno zažene predvajanje ali preklaplja med zagonom in zaustavitvijo ter barve urejevalnika.

16-bitni

Vsak korak ima noto, glasnost, val in učinek. Glasnost je od 0 do 7. 0 pomeni tiho.

Valovi:

- **0 trikotnik**
- **1 nagnjena žaga**
- **2 žagi**
- **3 kvadratne**
- **4 impulzi** – ozek impulz
- **5 orgel**
- **6 hrupa**
- **7-fazor**
- **8 PWM** – kvadrat, katerega širina se premika
- **9-kratni pulz 25 %**
- **10 sinus**
- **11 kovinski hrup**
- **12 FM zvon**
- **13 odtrgati**
- **14 superžaga**
- **15 PWM (C64)**

Učinki na posamezni noti:

- **0 noben**
- **1 diapozitiv** – višina tona in glasnost se premikata do naslednje note v tem koraku
- **2 vibrato** — nihanje višine tona
- **3. padec** — višina tona pade v tišino, običajni brcanje
- **4 postopno pojavljanje**
- **5 postopnega izginjanja**
- **6 arpeggio** — sprehodi to noto in naslednje tri, eden na vsakih 8 taktov. Pri TIC 8 ali manj, vsake 4 takte.
- **7-hitri arpeggio** – iste štiri note, vsake 4 takte ali vsaki 2, ko je TIC 8 ali manj.

Glava vzorca, nad opombami:

- **TIC** – število utripov na korak, od 1 do 255. 15 je 120 utripov na minuto. Vnesite številko ali jo povečajte za eno.
- **LOOP IN / OUT** – če je OUT nad 0, se vzorec ponavlja od IN do OUT. Če pustite OUT na 0, naslov postane LEN: IN označuje število predvajanih not, 0 pa pomeni celoten vzorec.
- **ŠUM** – celoten vzorec postane šum.
- **CRUSH** - živahna različica vsakega vala.
- **DTUNE** — 0 izklopljeno, 1 rahel refren, 2 močan refren.
- **RVERB** — odmev: 0 izklopljen, 1 kratek, 2 dolga.

- **DAMPN** — nizkoprepustni filter: 0 izklopljen, 1 šibek, 2 močan.
- **LEVEL** — glasnost celotnega vzorca, od 0 do 7.
- **SWING** — premešanje. Lihi koraki čakajo malo dlje, sodi koraki malo manj.
- **KORAKI** – koliko korakov vzorec uporablja: 32, 24, 16 ali 12.

### 8-bitni

Korak ima noto in val. Štirje valovi so kvadratni, trikotni, žagasti in šumni. Glava ima TIC, LOOP IN / OUT, FX in LEVEL. TIC je ena od 1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 16, 20, 24, 30, 40, 60. FX je en učinek za celoten vzorec: 0 brez, 1 vibrato, 2 arpeggio, 3 postopno zmanjševanje. LEVEL je od 0 do 3.

### 4-bitni

Korak je samo nota. Glava ima TIC, LOOP (vklopljeno ali izklopljeno), WAVE (isti štirje valovi) in OCT (v kateri oktavi se nahajajo note). TIC je ena od 4, 6, 8, 10, 12, 15, 20, 30.

### 2-bitni

Korak je prazen, boben, udarec ali stisnjen. Glava ima TIC in BASE. TIC je 8, 15, 16 ali 30. BASE določa, kako visoko sedita boben in udarec.

### 1-bitni

Korak je nota. TIC je 8, 15, 16 ali 30, iste štiri hitrosti kot 2-bitni.

### Pesmi

Zavihek ZVOK predstavlja odprto pesem. Osem stolpcev predstavlja kanale. Celica predstavlja ID vzorca. Če je 255 prazna, pomeni, da je kanal v tej vrstici tih. Stolpec X je ukaz: nadaljevanje, zaustavitev, začetek zanke ali konec zanke. Stop konča pesem po vrstici. Konec zanke skoči nazaj na zadnji začetek zanke.

Vrstica se konča, ko se konča skrajno levi vzorec z zanko LOOP OUT pri 0. Vzorci bolj desno te vrste ne podaljšajo: daljši vzorec se odreže, krajši pa se začne znova na svojem prvem koraku, dokler se vrsta ne konča. Vzorec z zanko LOOP OUT nad 0 se zanka med zanko IN in zanko OUT in ne konča vrste. Če se vsi vzorci v vrstici zankajo, vrsta tam ostane.

### Bližnjice

| Ključ | Dejanje |
|---|---|
| Klavirske tipke | Napišite opombo o koraku, izbranem z miško. |
| Gor / Dol | Spremenite vrednost pod kazalcem. |

| | |
|---|---|
| Vesolje | Predvajajte odprti SFX ali odprto pesem iz prve vrstice. Začnite/ustavite ali ponovno zaženite v nastavitvah UREJEVALNIKA SFX. |
| Ctrl+Z / Ctrl+Y | Razveljavi / ponovi. |
| Ctrl+C / Ctrl+X / Ctrl+V | Kopiraj, izreži, prilepi izbor. |

## PODATKI

Poimenovane tabele. Vsaka tabela je sestavljena iz vrstic in stolpcev. Iz kode `dget`, `dset`, `dstr`, `drows` in `dcols` uporabljajo te tabele. Številke vrstic in stolpcev v kodi se začnejo pri 0.

Vrste stolpcev, ki se ciklično spreminjajo v glavi:

- **bool** — en bit.
- **2** — dva bita.
- **4** — štirje biti.
- **8** — bajt.
- **16** — 16 bitov, za ID-je in večje številke.
- **spr** — ID spritea.
- **str** — besedilo, narisano s sistemsko pisavo.
- **str8** in **str10** – isto besedilo, narisano višje.
- **f16** in **f32** – polprecizni in polprecizni plovni utrip.

Ko je kazalec na glavi stolpca, Ctrl+drsenje spremeni vrsto tega stolpca. Navaden drsenje se premika po stolpcih ali vrsticah navzdol, kar koli izbere NASTAVITVE → UREJEVALNIK PODATKOV. Ctrl zamenja to smer, zato Ctrl+drsenje na glavi spremeni vrsto, ko je kolesce nastavljeno na stolpce.

S puščičnimi tipkami se premikate. S tipko Enter urejate celico. NASTAVITVE → UREJEVALNIK PODATKOV:

- **Kolo** – pomikanje po stolpcih ali vrsticah.
- **Barve** – mreža, glava in izbor.

## Sredstva

ASSETS uvozi in izvozi odprt del, ne celotnega projekta. Spriteji uporabljajo sprite, odprt v programu PAINT, na odprtem listu. Zemljevid je odprt zemljevid. SFX je odprt vzorec. Glasba je odprta pesem. Podatki so odprta tabela. Izjema je koda: vsak zavihek je zapisan in vsak zavihek je prebran nazaj.

| Vrsta | Datoteka | Kaj se premika |
|---|---|---|
| Špriti | `.png` | Odprti sprite. |

| | | |
|---|---|---|
| Zemljevid | `.p37map` | Odprti zemljevid. |
| Zvočni učinki | `.p37sfx` | Odprti vzorec. |
| Glasba | `.p37mus` | Odprta pesem. Številke vzorcev ostanejo takšne, kot so. Če potrebujete te zvoke, izvozite zvočne učinke ločeno. |
| Podatki | `.p37dat` | Odprta miza. |
| Koda | `.lua` | Vsak zavihek. |

## Nastavitve

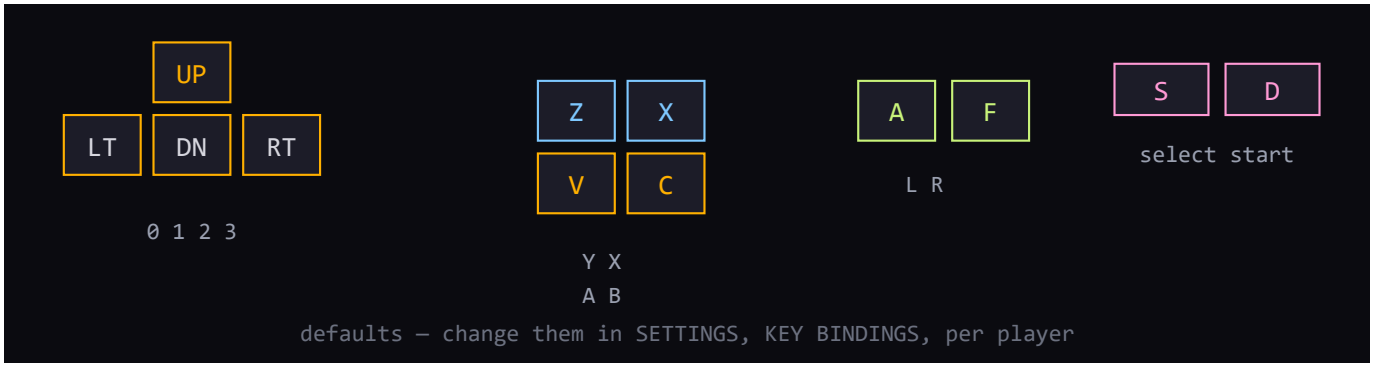
Korenski seznam:

| Stran | Kaj shranjuje |
|---|---|
| PROFILI | Shrani in naloži vsako nastavitev. Novo, preimenuje, prepiše, izbriše. Kopiraj ali prilepi profil kot besedilo. |
| Meni | Okno, tipke, barve menijev, neobvezno ozadje. |
| IGRALNA PLOŠČA | Predogled in okno za predvajanje. |
| UREJEVALEC KODE | Označevanje in namigi. |
| UREDNIK BARV | Čopič, paleta, pogled zemljevida. |
| UREDNIK ZNAČILNIH UČINKOV | Sintetizator in vnos not. |
| UREJEVALEC PODATKOV | Smer in barve koles. |
| JEZIK | Jezik menija in namigov. |
| DOVOLJENJA | Besedila licenc, vključno s pisavo Fusion Pixel in SDL3. |

GLOBALNO:

- **POSTAVITEV** — razporeditev plošč, zgoraj.
- **GLASNOST** — glavna raven, od 0 do 10.
- **VEZAVE TIPK** – tipke za vsakega od 8 igralcev. Pritisnite tipko, da jo zajamete. Navodila, ABXYLR, izberi.
- **Predstavitev** – preskoči nedejavne okvirje ali nariši vsak okvir.
- **Celozaslonski način** – brez roba.
- **Osveži** – število sličic na sekundo v oknu. Korak levo in desno: 60, 90, 120 in največ. Vnesite število do 1000. 0 pomeni brez omejitve.

- **Vsync** – počakajte na prikaz med okvirji.
- **Lestvica** — celoštevilski množitelj okna.
- **Shaderji** — CRT in filtri v urejevalniku, ne v igri.
- **Barva** – besedilo menija in označevanje.
- **Ozadje** – neobvezna datoteka PNG, obrezana na 128, za menijem. Z gumbom Počisti jo odstranite.
- **Kurzor urejevalnika** – oblika kazalca.
- **Miška** – plavajoči kazalec ali pripet na slikovne pike.
- **Hitrost uporabniškega vmesnika** – kako hitro se zadržana smer ponavlja po seznamu.
- **Raziskovalec** – seznami datotek se začnejo v korenski mapi računalnika ali v mapi exe.




Privzeta tipkovnica. A je potrditev v menijih. B je nazaj. V igri sta to gumba 4 in 5. Izberi (S) je gumb 10, ne bližnjica v meniju.

| Gumb | Vloga | Privzeti ključ |
|---|---|---|
| 0 | Levo | Puščica levo |
| 1 | Desno | Desna puščica |
| 2 | Gor | Puščica navzgor |
| 3 | Navpično | Puščica navzdol |
| 4 | A, potrdi | V |
| 5 | B, nazaj | C |
| 6 | X | X |
| 7 | Y | Ž |
| 8 | R | Ž |
| 9 | L | A |
| 10 | Izberite | S |

| | | |
|---|---|---|
| 11 | Začni | D |

## Zgraditi

Izberite platformo: Windows, Linux, Mac, Android, iOS ali splet.

- **Ime** — ime datoteke gradnje.
- **Avtor** – v sistemih Windows in Linux. To je ime mape za shranjene datoteke.
- **ID aplikacije Steam** – v sistemih Windows in Linux. Prazno pomeni, da Steamworks ni na voljo. Izpolnjen ID preslika `achieve(0)`...`achieve(15)` v dosežke Steam ACH_0...ACH_15.
- **Id** — identifikator paketa, kjer ga platforma zahteva.
- **Skrij kodo** – na prekrivnih elementih za namizje in mobilne naprave. Upočasni `peek` igralne kartuše. To ni ključavnica.
- **Celoten zaslon** – izvoženi predvajalnik se zažene v celozaslonskem načinu.
- **Senčilniki** — do tri plasti, vdelane v predvajalnik.
- **Prenesi .p37** – Samo za splet. Prikaže vnos za prenos v meniju za premor.
- **Ikona** – ploščica iz lista sprite: list, velikost, x, y. Ali pa obrežni naslov PNG.
- **Izvozi** – izberite mapo. V meniju se prikaže sporočilo GRADNJA SHRANJENA ali GRADNJA NI USPELA.

Naredite gradnjo z vključenim internetom. Vnos lahko anonimno vpraša strežnik, ali je zgoščena vrednost te gradnje znana. To preverjanje je tam iz dveh razlogov. Ščiti mene, avtorja PIX37, pred kopijami orodja, ki niso bile moje, da bi jih podaril. Ščiti tudi osebo, ki je igro ustvarila. Vzemanje igre nekoga drugega in njeno objavo kot svoje postaja običajno na trgu telefonov, kot sta Play (Android) ali iOS. Igra se še vedno zažene. Če je naprava brez povezave, uvod pove, da igra ni povezana. Ta vrstica je tudi namig igralcu, da kopija morda ni prišla iz zakonitega vira. Gre za majhno preverjanje, tako da če je vašo igro ukradel in prodal nekdo drug brez vašega soglasja, lahko to utemeljite v trgovini, kot so Play (Android), Steam ali itch.io. Strežnik shrani čas, ko je bil zgoščena vrednost prvič registrirana, poleg zgoščene vrednosti in ga kasneje ne spremeni. Zgodnejši čas je prvotna gradnja. Zgoščena vrednost, registrirana po tem, je poznejša kopija.

Želim si, da bi to preverjanje zaščitilo ljudi, ki ustvarjajo igre. Igrenski mehanizem že opravi veliko dela namesto njih in to je ena od teh stvari. Razumem, da je lahko sporno. Če se bo skupnost odločila, da je bolje opustiti ta mikro-DRM, bom to storil.

Stara igralna kartuša iz prejšnjega urejevalnika se lahko po spremembi jezika zavrne za zagon. V tem urejevalniku odprite projekt `.pix37` in izvozite novo datoteko `.p37`.

## Povežite se s PIX37.PARTY

POVEŽITE SE S PIX37.PARTY je znotraj aplikacije. V brskalniku ni strani za prijavo. Prijavite se s sejno vstopnico Steam ali z itch.io, z uporabo ploščice in miške. Storitev prikazuje edinstveno prikazno ime, dolgo od 2 do 16 znakov. Licenčni ključ in Steam ID nista prikazana drugim igralcem.

Po pogojih storitve so središče Priljubljene, Katalog, Lestvica, Igralci in vaš profil.

Razvijalska licenca je urednik. Vi ustvarite igro, jo naložite in pustite zvezdice. Igralna licenca je igralec. Srčki so od ljudi, ki igrajo na tekaški igri.

- **Katalog** – brskajte po igrah. Filtri zajemajo žanr (do tri), velikost zaslona, velikost igralne kartuše in ali nalaganje vključuje projekt `.pix37`.
- **Priljubljene** – igre, ki ste jih označili.
- **Nalaganje** – potrebna je razvijalska licenca. Pošljete `.p37` in lahko priložite projekt `.pix37`. `.p37` je zapakirana bajtna koda. Težko jo je brati in jo je mogoče namerno zakriti. `.pix37` je projektna datoteka, s katero dejansko delate: izvorna koda, slike in zvok v obliki, ki jo lahko odpre druga oseba in se iz nje uči. Če jo pošljete, lahko drugi ustvarjalci preučujejo igro namesto stisnjene igralne kartuše. Zraven spadajo naslov, avtor, žanri, naslovnica, ikona in imena dosežkov. Poslana številka RAM-a predstavlja vrh, ki ga je igralna kartuša dosegla.
- **Komentarji in dosežki** – na strani igre. Odklepanje dosežka, ko ste prijavljeni, se shrani v tem računu in na napravi.
- **Lestvica** – razvijalci, razvrščeni po zvezdicah ali srčkih. Vsaka vrstica prikazuje obe vrednosti. Odprite ime, da si ogledate igre tega avtorja, razvrščene po srčkih.
- **Igralci** – računi, razvrščeni po odklenjenih dosežkih.

# Koda

Jezik je majhna skripta v slogu Lua. Ta polovica se začne s praznim projektom, uči jezik in nato prikazuje kratek primer vsake funkcije.

## Prvi program

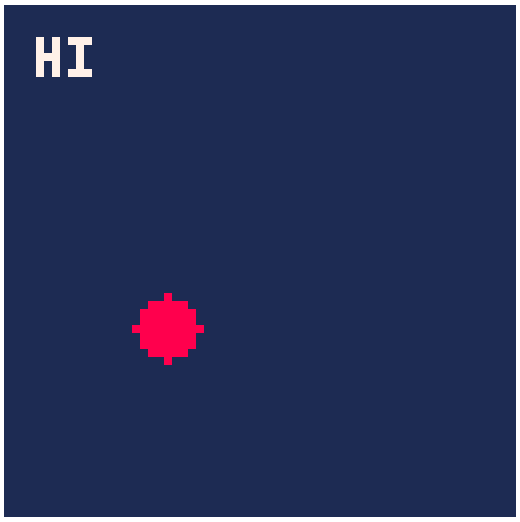
Ustvarite projekt z ločljivostjo 64 × 64, 4-bitnimi barvami. Odprite KODO. Zamenjajte jeziček z:

```
_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 jo zažene. S puščičnimi tipkami premikate krog. Escape ustavi igro in se vrne v urejevalnik.




Isti trije klici: `cls(1)` nariše temno modri zaslon, `print` nariše HI in `circfill` nariše roza krog.

## Kaj teče in kdaj

- `_init` se zažene enkrat.
- `_update` deluje s frekvenco 60 Hz. Tukaj postavite gibanje in gumbe.
- `_draw` slika. Brez argumenta fps slika tako hitro, kot zmoreta računalnik in monitor. `_draw(30)` omejuje slikanje na 30 fps in pusti logiko na 60.
- `_set_fps(n)` to omejitev kasneje spremeni. 0 jo počisti.

Imena niso občutljiva na velike in male črke; urejevalnik jih shrani z malimi črkami. `function` in `fun` sta ista ključna beseda.

## Jezik

### Osnove programiranja

Program je seznam vrstic, ki jih izvaja igralna kartuša. Vsak naslov najprej prikazuje obliko vrstice, nato pa majhen primer. Vrstica, ki se začne s `--`, je komentar: opomba za vas. Igra jo preskoči.

#### Spremenljivke

Spremenljivka je ime, ki vsebuje vrednost. Ime izberete sami. Kasnejše vrstice jo lahko preberejo in shranijo novo vrednost pod istim imenom.

Oblika je `name = value`. `local name = value` ohrani to ime znotraj trenutnega bloka, zato ga preostali del igralne kartuše ne more videti. Brez `local` je ime globalno in ga lahko vidi vsak zavihek.

```
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` pomeni, da ni vrednosti. `true` in `false` sta da in ne. Ime, ki ga še nikoli niste nastavili, se bere kot `nil`.

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

#### Mize

Tabela vsebuje veliko vrednosti pod enim imenom. Seznam uporablja številke. Prvi element je 1, ne 0. Zapis uporablja imena. Tabela lahko vsebuje drugo tabelo.

Oblika seznama: `{ value, value }`. Oblika zapisa: `{ name = value, name = value }`. Seznam zapisov je tabela, katere elementi so tabele.

```
-- 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
```

### Stanje in telo

Pogoj je vprašanje z odgovorom da ali ne, na primer `hp > 0` ali `btnd(4)`. Telo je skupina vrstic, ki se začnejo, ko vstopite v blok. `if` in vsaka zanka uporablja isto obliko: pogoj, nato telo in nato `end`. `repeat` se zaključi s `until` namesto s `end`.

### Če

`if` zažene svoje telo le, če je pogoj resničen. `elseif` poskusi z drugim pogojem. `else` se zažene, ko so bili vsi pogoji napačni.

Oblika: `if condition then`, nato telo in nato `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
```

### Zanke

Zanka ponavlja telo. `while` najprej preveri pogoj in se nadaljuje, dokler ta ostane resničen. `repeat` zažene telo, nato pa preveri, tako da se telo vedno izvede vsaj enkrat. `for` šteje. Vključena je zadnja številka. Brez tretje številke je korak 1, zato se telo ne izvede, če je začetek že mimo konca. Odštevanje nazaj zahteva negativni korak.

```
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
```

### Hoja po mizi

`all` obišče vsak element seznama, od prvega. `pairs` obišče vsako ime in vrednost, shranjeno pod tem imenom.

```
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
```

### Funkcije

Funkcija je poimenovan recept. Oblika je `function name(arguments)`, nato telo in nato `end`. Pokličete jo tako, da zapišete ime in oklepaje. `return` pošlje nazaj vrednost in jih lahko pošlje več. Argumenti so lokalni podatki te funkcije.

```
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` je funkcija, ki lahko vidi lokalne vrednosti okoli sebe. To je zaprtje. Tabela lahko vsebuje eno vrednost in jo pokličete s piko.

```
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
```

### Da, ne in izbira

`not` obrne pogoj. `and` je resničen le, če sta obe strani resnični. `or` je resničen, če je katera koli stran resnična. `a and b or c` izbere `b`, če je `a` resničen, sicer pa `c`. `cond ? a : b` je ista ideja v eni vrstici: `a`, ko je pogoj resničen, `b`, ko ni.

```
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)
```

### Matematika in besedilo

`+ - * /` so seštevanje, odštevanje, množenje in deljenje. `//` deli in ohrani celo število. `%` je ostanek. `^` je potenca. `..` zlepi dve besedili. `#t` je število elementov na seznamu. Primerjave so `==` (enako), `~=` ali `!=` (različno) in `< > <= >=`. `x += 1` sešteje 1 z x. Ista oblika velja za `-= *= /= ^= %=`.

```
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
```

### Kjer živi domačin

`local` traja, dokler `end` ne zapre svojega bloka. `if` / `end` je svoj blok. Nad vrstico, ki ga nastavi, lahko omenite lokalno vrednost, vendar je vrednost `nil`, dokler se ta vrstica ne zažene.

```
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
```

### Skakanje

`::label::` označuje vrstico. `goto label` skoči tja, vendar le znotraj iste funkcije.

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

## Spomin, skratka

`peek` in `poke` pregledujeta žive slikovne pike prek enega 16-bitnega naslova. Ne hranita druge kopije igralne kartuše. Naslovi od `0x5F00` do `0x5FFF` so registri: kamera, posnetek (samo za branje), barva risanja, vzorec polnjenja, tri preslikave palet in številke strani. Bajti na obeh straneh te luknje, teh 65280, predstavljajo okno slikovne pike.

Ko se zaslon prilega v teh 65280 bajtov, 64×64 in 128×128 pa, 256×256 pa ne, se zaslon začne na naslovu 0, vrstico za vrstico. Ostane le list spriteov, ki ga izbere `0x5F41` (0 je list 0). Če je ta list večji od preostalega prostora, `0x5F42` izbere stran in zemljevid ni v oknu. Če se list prilega, so na repu celice zemljevida iz zemljevida, ki ga izbere `0x5F43`, vsaka po en ali dva bajta, `0x5F44` pa zemljevid izpiše na strani, ko se ne prilega na rep.

Ko zaslon ne ustreza velikosti, je celotno okno ena stran s 65280 slikovnimi pikami. `poke(0x5F40, n)` obrne stran zaslona. `poke(0x5F41, n)` z n, ki je drugačen od 0, namesto tega prikaže list n, list 0 pa v tem načinu nima pogleda. `poke(0x5F43, n)` z n, ki je drugačen od 0, prikaže zemljevid n, ki prevlada nad listom. Zemljevid 0 v tem načinu je samo `mget` in `mset`.

`peek`, ki ne pristane na slikovni piki ali registru, vrne 0. `poke` tam ne naredi ničesar. Zastavice sprite, SFX in tabele DATA niso v tem oknu. Uporabite `fget`, `sfxget` in `dget`.

## Praktična uporaba

Vsaka funkcija je navedena na dnu te strani s praktičnimi uporabami funkcij PIX37. Povezave pod vsakim primerom odprejo več o funkcijah.



## Gladko gibanje

Igralni vložek velikosti 128 × 128, gledano od zgoraj. Sprite 1 je že narisan. Puščice pospešujejo v katero koli smer, trenje pa vas upočasni. Sprite se ovije: polovica, ki zapusti dno, je narisana na vrhu, enako pa velja tudi za ostale robove.

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



## Preverjanje, ali je naslednja poteza blokirana

Igralni vložek velikosti 64×64. Tla in desna stena sta že na zemljevidu, pobarvana s spritom 3, na tem spritu pa je nastavljena zastavica 0. Puščice premikajo sprit 1. Beseda na vrhu pravi "blokirano", ko bi se naslednji korak prekrival s polno ploščico, in "prosto", ko se ne bi.

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



## Robček z obračanjem

Pritisnite A. Brisanje se izriše s funkcijo flip(), eno črno vrstico naenkrat, namesto znotraj funkcije _draw(). Funkcija _draw se med izvajanjem ne moti, zato vsaka vrstica ostane na zaslonu.

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



## Tabela funkcij (razred)

Sprite 1 je že narisan. make_monster deluje kot razred. Vrne tabelo, update in draw pa sta metodi, ki ohranita x in y te pošasti. Številka na vrhu predstavlja, koliko je pošast prehodila.

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



## Štirje načini branja gumba

A je gumb 4. Držite ga in naboj narašča. Spustite ga in naboj se vrne na 0. Gor in dol se ponavljata, medtem ko držite gumb, tako da se vrstica menija premika. btnd je prvi okvir pritiska, ki ga ta igralna kartuša uporablja le kot začetek tega držanja.

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



## Streljaj z miško

Sprite 1 je že narisan na sredini. Bela pika je kazalec. Levi klik sproži slikovno piko proti njej. Kolesce spremeni hitrost, natisnjeno v kotu.

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



## Vnesite ime in ga potrdite

Pritisnite A, da odprete polje. Z gor in dol spreminjate utripajočo črko. A jo zapiše, X izbriše, Y je presledek in B potrdi. Vrstica nato izpiše vaše ime.

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



## Kamera na predvajalniku

Igralna kartuša velikosti 128 × 128. Sprite 1 se začne na sredini tal iz ploščic. Puščice se premikajo. Kamera ohranja sprite centriran. x in y v kotu se ne pomikata, ker camera() brez argumentov vrne pogled nazaj za HUD.

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



## Nekaj vrstic vizualnega romana

Vrstice so v jeziku PODATKI, v angleščini in poljščini. A se nadaljuje ali izbere ostane, ko je na zaslonu možnost. B izbere možnost, ki ostane, sicer pa zamenja jezik. X zamenja jezik, ko je prikazana možnost. Dialog uporablja pisavo velikosti 8px, zato se rišejo poljske črke.

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



## Slikajte s kukanje in pokanje

Igralna kartuša velikosti 64 × 64, zato je naslov 0 zgornji levi slikovni pik, naslednja vrstica pa se začne pri 64. Igralna kartuša zapisuje slikovne pike, kopira vrstico, izbriše vrstico in izpiše vrednosti, ki jih je prebrala nazaj.

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



## Seznam, ki ga lahko dodate

A sešteje število do 8. B odstrani zadnje. X odstrani prvo. Seznam se izpiše po zaslonu.

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



## Razstavljeno ime

Igralna kartuša razdeli ada,12,blue in izpiše ime, prvo črko, 12, in besedo blue.

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



## Zaigraj noto in jo nato spremeni

Vzorec 0 je dolg eno noto in vsaka ponovitev je enake glasnosti. A ga ponovno zaigra, ko ta nota konča. B zviša višino tona in zaigra to noto.

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



## Pesem, zapisana v igralni vložek

V vrstici 0 se ponavljata dva vzorca, po en na vsakem od prvih dveh kanalov. Vrstica ostane enaka 0. ch0 in ch1 sta nota, na kateri je vsak kanal. Urejanje vzorca, LEN in LOOP v igralni kartuši je način, da vidite, kaj ti kontrolniki počnejo na glasbeni plošči.

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



## Tri pošasti v DATA

Plošča PODATKI že ima stolpec z imenom in stolpec z močjo (hp), s sluzi, netopirjem in očesom. A škoduje sluzi. Številke na desni so njene moči (hp).

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



## Zapomni si rezultat

A doda točko in jo shrani. Zapri igro in ponovno odpri kartušo z igro: številka je še vedno tam. B jo pozabi.

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




## Drugo poglavje v drugi datoteki

Dve igralni kartuši si delita nalogo shranjevanja. 1. del: A označuje konec igre. 2. del, ki se odpre prvi, pravi, da je treba končati 1. del. Po tem se odpre 2. del, B pa izbriše shranjeno datoteko, tako da je mogoče 1. del znova igrati.

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



## En dosežek

Slika št. 8 je že narisana. Pritisnite tipko A desetkrat. V obvestilu se pojavi ime dosežka, slika št. 8 pa ostane, ker je odklenjena. Tipka B izbriše dosežek v lokalni datoteki in ponastavi števec.

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



## Soba, v katero lahko vstopiš

Igralna kartuša velikosti 256 × 256. Bela pika je kazalec. Meni je gostitelj, pridruži se ali seznam strežnikov. Gostitelj odpre sobo z imenom sprehod in začne igro. Pridruži se pomakne za naslov (številke, pika, dvopičje) in

B se poveže. Seznam strežnikov se premika gor in dol, A pa se pridruži vrsti. V igri puščice sprehajajo po spritu 1. En kovanec se deli: gostitelj ga premakne in ko ga pobere, se njegovemu rezultatu doda 1, nato pa se pojavi naslednji. Vsak sprite, vključno z vašim, prikazuje ta rezultat poleg sebe. Vogal prikazuje, koliko igralcev je notri, naslov LAN in javni naslov gostitelja. B odide: vi in ostali se vrnete v meni, naslednji gostitelj pa začne novo sobo z rezultatom 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`



### Dialog, ki čaka

A prikaže naslednjo vrstico. Funkcija se pri vsakem izidu ustavi, dokler ponovno ne pritisnete A.

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



### Počakajte pol sekunde

Beseda se po približno pol sekunde spremeni iz ... v hi. Zaslon se med začasno ustavitvijo korutine še naprej riše.

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



### Krogi, ki prihajajo eden za drugim

Z vrha kaplja pet krogov, dva okvirja narazen, in nato pade.

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



### Dva okvirja iz ure

Sprite 1 in sprite 2 sta dva okvirja istega znaka. Slika se zamenja štirikrat na sekundo.

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

## Vsaka funkcija

Argumenti v oglatih oklepajih so neobvezni. Indeksi tabel in nizov se začnejo pri 1, razen vrstic in stolpcev DATA, ki se začnejo pri 0. Barve so indeksi palet.

### Kazalo


`_risanje`
`_init`
`_mp`
`_mp_fail`
`_mp_join`
`_mp_leave`
`_mp_update`
`_set_fps`
`_posodobitev`

abs
achclear
achdef
doseči
dosežen
dodati
vse
trditi
atan2
bend
bnot
bor
btn
btnd
btnp
btnu
bxor
fotoaparat
ceil
hr
circ
circfill
clickl
clickm
clickr
posnetek
cls
soustvariti
barva
coresum
cos
costatus
štetje
kazalec
dcols
del
delikatesa
delsave
dget
drows
dset
dstr
fget
fillp
flip
flr
foreach
fset
info
ipairs
ključavnica
keyon
keystat
keystr
linija
obremenitev
dnevnik
lshr
zemljevid
maks
memcpy
memset
mget
sredina
min
miško
mset
glasba
musicstat
nannounce
ncall
naslednji
nhost
pridruži se
nkb
nkbstr
nlan
nleave
nlist
nposlušaj
opomba
nplayers
nsend
nstun
red
ovalne
ovalno polnilo
paket
parov
prijatelj
palrgb
palt
pcall
pokukati
pokukati2
pokukati4
uspešnost
pget
poke
poke2
poke4
tiskanje
natisni10
natisni8
pset
rect
rectfill
rnd
rotl
rotr
pravilno
rrectfill
shraniti
savesslot
zaslon
drsenje
izberite
setmap
setpalette
setspr
settex
setturn
sfx

| | | |
|---|---|---|
| sfxget | greh | stat |
| sfxhdr | songget | sub |
| sfxset | nabor pesmi | čas |
| sfxstat | razdeliti | tline |
| sget | spr | tonum |
| sgn | sqrt | tostr |
| shl | srand | razpakirati |
| showstats | sset | whoami |
| shr | sspr | donos |

## Življenjski cikel

### _init

`_init() ... end`

Zažene se enkrat ob zagonu igralne kartuše. Koda, ki se nahaja zunaj katerega koli kavlja, se prav tako zloži v _init.

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

### _update

`_update() ... end`

Logika igre. Klicano pri 60 Hz, tudi ko je _draw omejen ali je gostitelj počasnejši. Gumbi za branje so tukaj.

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

## _draw

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

Nariše en okvir. Brez argumenta ali z 0 se izvaja tako pogosto, kot ga prikaže gostitelj. Podajte n (1..999), da omejite risanje na n okvirjev na sekundo. _update ostane pri 60 Hz.

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

## _set_fps

```
_set_fps(n)
```

Spremeni zgornjo mejo _draw med delovanjem igre. 0 pomeni neomejeno. n je omejen na 0,999.

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

## flip

```
flip()
```

Predstavi zaslon in se premakne do naslednjega okvirja. Uporabite ga le, če rišete svojo lastno zanko okvirja. Ne čakajte po njem.

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

## time

```
time()
```

Sekunde od _init.

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

## Vnos

### btn

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

1, medtem ko držimo gumb b, sicer 0. p je igralec, 0..7. Brez argumentov vrne bitno masko igralca 0. Gumbi: 0 levo, 1 desno, 2 gor, 3 dol, 4 A, 5 B, 6 X, 7 Y, 8 R, 9 L, 10 izberi, 11 začni.

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

### btnd

`btnd(b, [p])`

Res je samo v okvirju _update, ko je gumb b pritisnjen.

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

### btnp

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

True ob pritisku, nato ponovno med držanjem: 30 tikov zakasnitve, nato vsakih 8 tikov. Če gumb ni pritisnjen, vrne masko.

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

### btnu

`btnu(b, [p])`

Res je, da se gumb sprosti le na okvirju.

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

### mouse

`mouse()`

Vrne miško x in y v slikovnih pikah igre.

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

### clickl

`clickl([p])`

Levi gumb miške. clickl() je pritisnjen. clickl(1) je rob pritiska.

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

### clickr

`clickr([p])`

Desni gumb miške. Enako enoargumentno rob kot clickl.

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

### clickm

`clickm([p])`

Srednji gumb ali klik drsnega kolesca.

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

## scroll

`scroll()`

Gibanje kolesca miške v tem okvirju. Pozitivno in negativno je odvisno od smeri vrtenja kolesca.

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

## keyon

`keyon()`

Odpre besedilno polje, ki ga upravljajo gumbi. S tipkama gor in dol spreminjate črko od A do Ž. A napiše to črko. X izbriše. Y je presledek. B potrdi in zapre polje. Tipkovnica vanj ne vnaša podatkov.

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

## keyoff

`keyoff()`

Skrije besedilno polje.

```
keyoff()
```

## keystr

`keystr([s])`

Besedilo v polju, nato 1 na okvirju B potrdi, nato pa se črka gor in dol spreminjata. Ta črka je prazna, ko se polje zapre. keystr("") izbriše besedilo.

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

### keystat

```
keystat()
```

1, medtem ko je besedilno polje odprto, sicer 0.

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

## Risanje

### screen

```
screen()
```

Trenutna velikost zaslona igralne kartuše v slikovnih pikah. Vrne širino, nato višino. To je ločljivost igre, ne okna urejevalnika.

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

### cls

```
cls([col])
```

Počisti zaslon. Privzeta barva je 0.

```
cls(1)
```

### color

```
color(col)
```

Določi barvo risanja, ki se uporabi, ko kasnejši klic izpusti barvo. Brez argumenta vrne trenutno barvo.

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

### camera

`camera([x, y])`

Vsak klic risanja premakne vrednost. camera() se ponastavi na 0, 0.

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

### clip

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

Omeji risanje na pravokotnik. Peti argument, ki ni enak nič, se seka s trenutnim posnetkom. clip() obnovi celoten zaslon. Peek registrov posnetka je samo za branje.

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

### cursor

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

Določi, kam gre naslednji izpis brez koordinat, in po izbiri barvo. cursor() vrne x, y.

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

### pset

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

Nastavi en slikovni pik zaslona. Šteje se v statistiko slikovnih pik.

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

### pget

```
pget(x, y)
```

Barva slikovne pike zaslona pri x, y.

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

### print

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

Nariše besedilo velikosti 3×5 (UTF-8, obrezano iz pisave Fusion). Brez x in y se izpiše na mestu kazalca in premakne v naslednjo vrstico.

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

### log

```
log(...)
```

Argumente, ločene s presledki, doda v besedilni medpomnilnik. pix37_run izpiše ta medpomnilnik na stdout. Ne riše.

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

### print8

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

Enako kot pri tiskanju, s pisavo Fusion velikosti 8 px.

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

### print10

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

Enako kot pri tiskanju, s pisavo Fusion velikosti 10 px.

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

### line

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

Vrstica od x0,y0 do x1,y1. Dva argumenta se nadaljujeta od prejšnjega konca. line() ponastavi to verigo.

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

### circ

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

Obris kroga. Polmer je v slikovnih pikah.

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

### circfill

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

Polnjen krog.

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

### oval

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

Obris elipse znotraj omejevalnega okvirja. Vogali so vključujoči.

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

### ovalfill

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

Zapolnjena elipsa v omejevalnem okviru.

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

### rect

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

Obris pravokotnika. Vogali so vključujoči.

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

### rectfill

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

Zapolnjen pravokotnik.

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

### rrect

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

Obris zaobljenega pravokotnika. x in y sta zgornji levi kot. w in h sta velikosti v slikovnih pikah; obe morata biti večji od 0. r je polmer vogala in je privzeto nastavljen na 0. Vpet je na polovico krajše stranice.

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

### rrectfill

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

Zapolnjen zaobljen pravokotnik. Isti argumenti kot pri funkciji rrect.

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

### fillp

```
fillp([p])
```

16-bitni vzorec polnjenja za pozneje zapolnjene oblike. 0 je polna črka. fillp() vrne trenutni vzorec. Biti se premikajo po vzorcu 4×4.

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

### pal

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

Preslika barvo c0 v c1. način 0 je paleta za risanje, 1 paleta za prikaz, 2 paleta za polnjenje. pal() ponastavi.

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

### palrgb

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

Nastavi en vnos v paleto na RGB, vsak kanal 0 do 255.

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

### palt

`palt([col, t])`

Prosojnost. Funkcija palt(col) ali palt(col, true) naredi barvo prozorno. Funkcija palt(col, false) jo naredi trdno. Funkcija palt() naredi vse barve neprozorne.

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

### setpalette

`setpalette(n)`

Izbere banko palet n. setpalette() vrne trenutno banko.

```
setpalette(1)
```

### tline

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

Nariše črto na zaslonu, medtem ko vzorči slikovne pike zemljevida iz položaja zemljevida mx, my. mdx, mdy postopno vzorči zemljevid vzdolž črte.

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

## Sprite in zemljevidi

### spr

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

Nariše sprite n pri x in y. w in h sta velikosti v slikovnih pikah, privzeto 8. fx in fy se obrneta vodoravno in navpično, če sta različni od nič. tex je zavihek lista sprite. rot je kot v obratih (0..1), enaka smer kot sin in cos. 0 pusti sprite nerotiran.

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

### sspr

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

Kopira pravokotnik lista sprite na zaslon, pomanjšan na dw za dh. Privzete nastavitve ohranijo velikost izvorne kode.

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

### sget

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

Barva enega teksela na listu spriteov.

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

### sset

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

Zapiše en teksel na list spriteov.

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

### fget

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

Zastavice Sprite. fget(n) vrne bajt zastavice. fget(n, f) je true, ko je nastavljen bit zastavice f (0..7).

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

### fset

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

Nastavi zastavice spriteov. fset(n, v) zapiše cel bajt. fset(n, f, v) nastavi en bit.

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

### setspr

```
setspr(n)
```

Izbere aktivni list sprite. setspr() vrne trenutni list. Isti bank kot settex.

```
setspr(1)
```

### settex

```
settex(n)
```

Vzdevek uporabnika setspr.

```
settex(0)
```

### map

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

Nariše zemljevid. Celici cx, cy zemljevida se narišeta na zaslonu sx, sy. cw, ch 0 pomenita celoten zemljevid. Plast se sprejme in prezre. tex je list s ploščicami.

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

### mget

```
mget(cx, cy)
```

ID spritea v celici zemljevida cx, cy.

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

### mset

```
mset(cx, cy, n)
```

Zapiše sprite n v celico zemljevida cx, cy.

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

### setmap

```
setmap(n)
```

Izbere zemljevid n. setmap() vrne trenutni zemljevid. Zemljevid 0 je pravi zemljevid.

```
setmap(1)
```

## Spomin

### peek

```
peek(addr)
```

Prebere en bajt iz 64 KB okna. Okno je zaslon v živo, nato slikovni elementi sprite, nato pa preslika celice, ko se zaslon prilega 65280 bajtom. Večji zasloni so ostranjeni s poke(0x5F40, page). Naslovi, ki niso preslikani, vrnejo 0. Zapisano je tudi kot @addr.

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

### poke

`poke(addr, ...)`

Zapiše bajte na naslov. Dodatni argumenti zapišejo naslednje bajte. Nepreslikani naslovi se prezrejo. Zapiše se tudi $naslov, vrednost. Drkanje po zaslonu se šteje kot pisanje piksla.

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

### peek2

`peek2(addr)`

Nepredznačena 16-bitna vrednost, little-endian, na naslovu.

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

### poke2

`poke2(addr, val)`

Zapiše nepredznačeno 16-bitno vrednost, po vrstnem redu od majhnega do zadnjega reda (little-endian).

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

### peek4

`peek4(addr)`

Nepredznačena 32-bitna vrednost, little-endian.

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

### poke4

```
poke4(addr, val)
```

Zapiše nepredznačeno 32-bitno vrednost, v vrstnem redu od majhnega do zadnjega reda (little-endian).

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

### memcpy

```
memcpy(dst, src, len)
```

Kopira len bajtov znotraj istega 64 KB okna.

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

### memset

```
memset(dest, val, len)
```

Zapolni len bajtov, začenši pri cilju, z vrednostjo.

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

## Matematika

### abs

```
abs(x)
```

Absolutna vrednost.

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

### flr

`flr(x)`

Zaokroži proti minus neskončnosti.

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

### ceil

`ceil(x)`

Zaokroži proti plus neskončnosti.

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

### max

`max(a, b)`

Večja od obeh vrednosti.

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

### min

`min(a, b)`

Manjša od obeh vrednosti.

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

### mid

`mid(a, b, c)`

Srednja od treh vrednosti. Uporabno kot sponka.

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

### sgn

`sgn(x)`

Predznak x: -1, 0 ali 1.

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

### sqrt

`sqrt(x)`

Kvadratni koren. Negativni argument ni število.

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

### sin

`sin(x)`

Sinus x je v obratih: 0, 0,25, 0,5, 0,75 obkroži krog enkrat. Ne gre za radiane.

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

### cos

`cos(x)`

Kosinus ima posledično enak dogovor kot sin.

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

### atan2

`atan2(dx, dy)`

Kot vektorja dx, dy se spreminja od 0 do 1.

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

## rnd

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

Naključno število v [0, x). Privzeto je x 1, zato je rnd() enak 0..1. Za kocko je flr(rnd(6))+1.

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

## srand

```
srand(seed)
```

Dodaje semena generatorju naključnih vrednosti, tako da se lahko poznejše zaporedje rnd ponovi.

```
srand(1)
```

## band

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

Bitni IN. Zapisano tudi kot a in b.

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

## bor

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

Bitni ALI. Zapisano tudi kot | b.

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

## bxor

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

Bitni XOR. Zapisano tudi kot ^^ b.

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

### bnot

`bnot(a)`

Bitno NE. Zapisano tudi ~a.

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

### shl

`shl(a, n)`

Premik levo. Zapisano tudi kot << n.

### shr

`shr(a, n)`

Aritmetični premik v desno. Zapisano tudi kot >> n.

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

### lshr

`lshr(a, n)`

Logični premik desno. Zapisano tudi kot >>> n.

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

### rotl

`rotl(a, n)`

Zavrtite bite v levo. Zapisano je tudi kot <<> n.

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

### rotr

```
rotr(a, n)
```

Zavrtite bite v desno. Zapisano je tudi kot >>< n.

## Tabele in vrednosti

### add

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

Doda v tabeli t. Če je podan i, vstavi na ta indeks, ki temelji na 1.

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

### del

```
del(t, v)
```

Odstrani prvi vnos, ki je enak v.

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

### deli

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

Odstrani indeks i in vrne to vrednost. Če ni indeksa i, odstrani zadnji vnos.

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

### all

```
all(t)
```

Iterator čez del polja, po vrstnem redu. Luknje ustavijo sprehod tako, kot se konča polje, ki temelji na 1.

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

### foreach

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

Za vsak element tabele pokliče fn(v).

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

### pairs

```
pairs(t)
```

Iterator nad vsakim ključem in vrednostjo, vključno z imenovanimi polji.

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

### ipairs

```
ipairs(t)
```

Iterator čez del polja z indeksom, ki temelji na 1.

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

### next

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

Vrne naslednji ključ in vrednost po k. next(t) se začne na začetku. Uporabljajo ga pari.

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

### count

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

Dolžina dela polja. Če je podan v, prešteje, koliko vnosov je enakih v.

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

### pack

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

Argumente zapakira v novo tabelo.

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

### unpack

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

Vrne vrednosti od t[i] do t[j] kot ločene vrednosti. Privzete vrednosti pokrivajo celoten del tabele.

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

### select

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

select("#", ...) vrne, koliko argumentov sledi. select(i, ...) vrne argumente od i naprej.

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

### assert

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

Ustavi se z napako, ko je c false ali nil. msg nadomesti privzeto besedilo.

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

### pcall

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

Kliče f. Ob uspehu vrne true in rezultat. Ob napaki vrne false in besedilo napake.

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

## Strune

### chr

```
chr(n)
```

En znak iz bajtne kode 0..255.

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

### ord

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

Bajtne kode s od indeksa i do j. Indeksi temeljijo na 1. Privzeto je nastavljen na prvi znak.

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

### sub

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

Podniz od i do j, ki temelji na 1. Izpustite j, da vzamete rep.

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

### split

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

Razdeli s v tabelo. sep je privzeto vejica. Število sep je širina bloka. conv pretvori kose s tonum, ko je ta true.

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

### tonum

```
tonum(s, [base])
```

Razčleni število. Osnova je lahko 2 ... 36. Vrne nič, če besedilo ni število.

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

### tostr

```
tostr(v, [digits])
```

Spremeni vrednost v besedilo. Funkcija digits nastavi decimalna mesta za števila.

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

## Zvok

## sfx

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

Predvaja vzorec n in vrne kanal. Če izpustite ch ali podate -1, je izbran prvi prosti kanal. ch -2 ustavi vzorec n na vsakem kanalu. n -1 ustavi kanal ali vsak kanal, če je ch izpuščen. n -2 omogoči, da se vzorec zanke konča namesto ponavljanja. sfx() brez argumentov vrne -1.

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

## sfxget

```
sfxget(n, step)
```

Vrne tabelo z enim korakom vzorca: [1] višina tona, [2] val, [3] glasnost, [4] učinek. Višina tona je 0..63 (C-0 do D#5). Val je 0..15. Glasnost je 0..7, 0 pa pomeni tišino. Učinek je 0..7.

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

## sfxset

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

Zapiše en korak vzorca. Zahtevanih je vseh šest argumentov.

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

## sfxhdr

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

Glava vzorca. sfxhdr(n) vrne tabelo glave. sfxhdr(n, key, val) nastavi eno polje: speed, loop_in, loop_out, level, noise, crush, dtune, rverb, dampn, swing, steps. Številke od 0 do 10 izberejo ista polja v tem vrstnem redu. speed je TIC, število udarcev na korak. Višja je počasnejša. 15 je 120 utripov na minuto. 16-bitni sistem sprejema 1 do 255. Ožji sintetizatorji se pripnejo na svoj seznam TIC.

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

## sfxstat

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

ID vzorca se predvaja na kanalu ch ali -1. Privzeti kanal je 0.

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

## note

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

Predvaja eno noto brez vzorca. Višina tona 0..63, val 0..15, glasnost 0..7, učinek 0..7. Čas predvajanja je enak časom predvajanja, v enakih 120 Hz taktih kot TIC. Če izpustite takte, traja 15, en privzeti korak. Učinek se predvaja v tem času, nato pa se kanal utiša.

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

## music

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

Začne predvajanje pesmi n od začetka. music(-1) ustavi predvajanje. Klic funkcije music z ID-jem, ki se že predvaja, jo pusti v teku. music() vrne ID predvajane pesmi ali -1. Fade je v milisekundah: pesem se pojavi ali ugasne, ko je n -1. Mask biti, ki so nastavljeni, so kanali, uporabljeni za pesem. 0 pomeni vseh osem. Ostali kanali ostanejo prosti za zvočne učinke.

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

### musicstat

```
musicstat()
```

Tabela predvajane pesmi. song in row sta ID pesmi in vrstica oziroma -1, ko se nič ne predvaja. step in pos imata osem števil. step[1] je kanal 0, step[8] je kanal 7. step je indeks note oziroma -1, ko je ta kanal tih. pos je, kako daleč je nota šla, od 0 do 1. tick je, kolikokrat je ta vrstica že predvajana. played je, koliko vrstic se je začelo od music(), vključno s tisto, ki se predvaja.

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

### songget

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

ID vzorca je shranjen v celici pesmi. ch je 0..7. 255 pomeni, da je celica prazna. To je aktivni klic; besedilo namiga kode še vedno prikazuje starejši vrstni red argumentov.

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

### songset

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

Zapiše ID vzorca v celico pesmi. Za brisanje celice uporabite 255.

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

### stat

```
stat(n)
```

Samo zvok. 16..23 je ID zvočnih učinkov na kanalu n-16. 24..31 je glasbeni vzorec na kanalu n-24. Vsak drug n vrne 0. Časovni okvir in RAM sta zmogljivost in informacije, ne statistika.

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

## Podatkovne tabele

### dget

`dget(t, row, col)`

Prebere celico podatkovne tabele t iz plošče PODATKI. Vrstica in stolpec se začneta pri 0.

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

### dset

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

Zapiše celico. Za besedilni stolpec uporabite niz.

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

### dstr

`dstr(t, row, col)`

Celica kot besedilo. Številke so pretvorjene.

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

### drows

`drows(t)`

Koliko vrstic ima tabela t.

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

### dcols

```
dcols(t)
```

Koliko stolpcev ima tabela t.

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

## Shranjevanje in dosežki

### save

```
save(k, v)
```

Shrani v pod ključem k v datoteki za shranjevanje. v je lahko število, niz ali tabela teh. Datoteka je stisnjena.

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

### load

```
load(k)
```

Prebere ključ k. Vrne nič, če ključ manjka.

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

### delsave

```
delsave(k)
```

Izbriše tipko k iz shranjenega stanja.

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

### saveslot

```
saveslot([name])
```

Izbere datoteko za shranjevanje v mapi avtorja. Če ime ni določeno, je reža naslov igre. Vrne 1, če ta datoteka že obstaja.

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

### achieve

```
achieve(i)
```

Odklene dosežek i, od 0 do 15. Shranjeno lokalno. Gradnja Steam prav tako nastavi ACH_i. Seja kataloga to sporoči igralčevemu računu.

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

### achieved

```
achieved(i)
```

1, ko je dosežek i že odklenjen.

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

### achdef

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

Besedilo za dogodek i, prikazano 15 sekund od dna zaslona. spr je neobvezna ploščica velikosti 8×8, tex njen list.

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

### achclear

```
achclear(i)
```

Izbriše dosežek i v lokalni datoteki. Steam ACH_i, ki je bil že nastavljen, ostane nastavljen.

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

## Omrežje

### whoami

```
whoami()
```

Vaš ID igralca, 0..15, medtem ko je seja končana. Brez povezave je -1.

```
id = whoami()
```

### nhost

```
nhost()
```

1, če je ta stroj gostitelj, sicer 0.

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

### nplayers

```
nplayers()
```

Koliko živih ID-jev obstaja v 0..15.

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

### nlisten

`nlisten()`

Začne gostiti UDP in vrne LAN naslov kot ip:port.

```
addr = nlisten()
```

### njoin

`njoin(addr)`

Pridruži se ip ali ip:port.

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

### nleave

`nleave()`

Zapusti sejo. Vrstniki vidijo _mp_leave z razlogom 0.

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

### nsend

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

Pošlje sporočilo z izgubo. Ime je dogodek. Sledijo do štiri dodatne vrednosti. p od 0 do 15 šepeta temu vrstniku; izpustite p za pošiljanje vsem.

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

### ncall

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

Enaka oblika kot nsend, vendar se sporočilo poskuša ponovno poslati, dokler ni potrjeno. Trije zaporedni neuspehi prekinejo sejo.

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

### nlan

```
nlan()
```

Trenutna LAN IP:vrata po nlisten.

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

### nstun

```
nstun()
```

Vaš javni IP:port iz STUN-a. Gost iz drugega omrežja se mu lahko poskusi pridružiti.

```
pub = nstun()
```

### setturn

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

Rezervna možnost TURN, ko neposredna pot ne uspe. Funkcija setturn() s praznimi argumenti jo izklopi. Naslov je gostitelj:vrata.

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

### nlist

`nlist(game)`

Tabela odprtih sej za to ime igre. Vsak vnos ima naslov, igralce in ime.

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

### nannounce

`nannounce(game)`

Srčni utrip, zato se ta gostitelj prikaže na seznamu neaktivnih za igro. Med gostiteljstvom ga pokličite s časovnikom.

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

### nkb

`nkb([on])`

Polje za naslov. nkb(1) ga odpre, nkb(0) ga skrije, nkb() pa vam pove, če je odprto. Z gumboma gor in dol spremenite znak, 0–9 ali piko ali dvopičje. A ga zapiše. X ga izbriše. Tipkovnica vanj ne vtipka.

```
nkb(1)
```

### nkbstr

`nkbstr([s])`

Besedilo v naslovnem polju, nato pa se znaki navzgor in navzdol spreminjajo. nkbstr("") počisti besedilo.

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

## _mp

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

To funkcijo napišete. Zažene se, ko prispe sporočilo. from je ID pošiljatelja. name je dogodek. Sledijo do štiri vrednosti.

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

## _mp_join

`_mp_join(p)`

Napišete to funkcijo. Vrstnik je prejel ID p. Ta se sproži tudi za vas; whoami() je ta ID.

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

## _mp_leave

`_mp_leave(p, why)`

Napišete to funkcijo. Vrstnik p je odšel. Zakaj je 0 normalen odhod. Zakaj je 1 prekinjena povezava.

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

### _mp_update

`_mp_update()`

To funkcijo napišete. Izvaja se vsakih 15 okvirjev _update, približno 4 Hz. Dober kraj za nsend pozicije.

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

### _mp_fail

`_mp_fail()`

Napišete to funkcijo. Seja je mrtva: gostitelj je izginil ali pa je ncall trikrat spodletel.

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

## Korutine

### cocreate

`cocreate(f)`

Iz funkcije f ustvari korutino. Ne zažene se, dokler ni izvedena korutina.

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

### coresume

`coresume(c, ...)`

Izvaja korutino c, dokler ne vrne vrednosti ali rezultata. V primeru uspeha: true in vrne vrednosti. V primeru napake: false in napaka.

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

### costatus

```
costatus(c)
```

"suspendiran", "deluje" ali "mrtv".

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

### yield

```
yield(...)
```

Vrne nadzor tistemu, ki je poklical coresume. Vrednosti postanejo rezultati tega coresumea.

```
yield(1)
```

## Mera

### info

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

Nariše HUD v živo v _draw: fps (povprečje 1 sekunda), upd, drw, px, max, ram, peak, ramc in net, ko je seja zagnana. info() uporablja kazalec za tiskanje. info(col) nastavi samo barvo. info(x, y) ali info(x, y, col) jo postavi. Številke uporabljajo B, k ali m, da se prilegajo majhnemu zaslonu. Sam klic se ne šteje v px.

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

### showstats

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

Isti HUD kot info, samodejno narisan pri 0, 0 po _draw, ignoriranje kamere in posnetka. showstats(0) ga skrije. showstats(1) ali showstats() ga prikaže.

```
showstats(1)
```

## perf

`perf(n)`

Ena številka HUD, da si jo lahko sam narišeš. 0 fps, 1 upd, 2 draw, 3 px, 4 max (najtežji okvir to sekundo), 5 ram, 6 ramc (ram plus spakirana igralna kartuša), 7 peak ram od začetka igralne kartuše.

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