# PIX37 rokasgrāmata

Redaktors, pēc tam kods. Grāmatas bezsaistes kopija vietnē pix37.party/docs.

# Redaktors

PIX37 DEV ir redaktors. Šeit jūs zīmējat, rakstāt un klausāties spēli, pēc tam eksportējat spēles kasetni vai atsevišķu versiju. Tā var arī atskaņot spēles kasetni un atvērt PIX37.PARTY, lai lejupielādētu spēļu kasetnes un izmantotu katalogu. PIX37 DEV EDUCATIONAL ir tas pats redaktors, un tas ir bezmaksas. Tam nav versiju, spēļu kasetņu eksportēšanas un nav piekļuves PIX37.PARTY. PIX37 Console ir atskaņotājs. Tā atskaņo `.p37` spēļu kasetnes, un jūs atverat PIX37.PARTY, lai lejupielādētu spēļu kasetnes un pārlūkotu katalogu.

## Projekti un spēļu kasetnes

`.pix37` fails ir projekts: avota cilnes, спрайtu lapas, kartes, skaņa un datu tabulas. `.p37` fails ir spēles kārtridžs, kuru spēlētājs izmanto: blīvi iesaiņots baitkods.

Saglabātie faili un sasniegumi atrodas autora mapē blakus lietotnes datiem. Operētājsistēmā Windows tas izskatās šādi: `C:\\Users\\Ada\\AppData\\Roaming\\PIX37\\Mira\\Pong.p37s` saglabāšanai un `C:\\Users\\Ada\\AppData\\Roaming\\PIX37\\Mira\\Pong.ach` sasniegumiem. Operētājsistēmā Linux tie paši faili ir `~/.local/share/PIX37/Mira/Pong.p37s` un `~/.local/share/PIX37/Mira/Pong.ach` . Ada ir konts datorā. Mira ir autora vārds, un šis nosaukums ir mape. Pong ir saglabāšanas slots, kura noklusējuma iestatījums ir spēles nosaukums. `save` ieraksta `.p37s` failu. Pirmais atbloķētais sasniegums ieraksta `.ach` failu. Ja faili jau ir tur, skrējējs tos nolasa, startējot spēles kasetni.

## Izstrādātāju izvēlne

Escape atver izstrādātāja izvēlni. Escape taustiņš spēles darbības laikā atgriežas redaktorā. Apstipriniet ar Enter vai pogu A (pēc noklusējuma V). Atpakaļ ir poga B (pēc noklusējuma C).

Ja projekts nav ielādēts, izvēlne ir šāda: IZVEIDOT PROJEKTU, IELĀDĒT PROJEKTU, VEIDOT, IELĀDĒT spēles kasetni, IZVEIDOT SAVIENOJUMU AR PIX37.PARTY, IESTATĪJUMI.

Kad projekts ir ielādēts, tas ir: SAGLABĀT PROJEKTU, IELĀDĒT PROJEKTU, IZVEIDOT PROJEKTU, AKTĪVI, IZVEIDOT, IELĀDĒT spēles kasetni, IZVEIDOT SAVIENOJUMU AR PIX37.PARTY, IESTATĪJUMI.

Ctrl+S saglabā atvērto projektu. Ctrl+R to palaiž.

## Izveidot projektu

Pirms projekta izveides sākšanas CREATE PROJECT pieprasa spēles kaseti.

- **VIDEO** — 1 BITS (2 krāsas), 2 BITI (4), 4 BITI (16), 8 BITI (256).
- **AUDIO** — sintezatora parauga dziļums: 16, 8, 4, 2 vai 1 bits.
- **EKRĀNS** — iepriekš iestatītie izmēri ietver 64 × 64, 128 × 128, 160 × 90, 256 × 256 un 320 × 180. PIELĀGOTS izmērs ļauj izvēlēties jebkuru izmēru no 1 × 1 līdz 512 × 512.
- **PALETES** — krāsu bankas. Pirmā banka sākas ar 16 krāsu komplektu, kas parādīts zemāk. LOAD PALETTE lieto iebūvētu komplektu: PICO-8 16, PICO-8 32, Game Boy, 1-bit, CGA, Sweetie-16 un garāku sarakstu no Lospec (Resurrect 64, Apollo, Endesga 32 un 64, CC-29, SLSO8, Pear36, Oil 6, Duel, AAP-64, Lost Century un citi). Varat arī ielādēt paletes failu vai ielādēt savu PNG un izņemt paleti no šī attēla.




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

Krāsu indeksi, ko izmanto `cls` , `pset` , `spr` un pārējie. Paletes redaktorā var pārkrāsot jebkuru indeksu.

## Pieci paneļi

Redaktors ir GAME, PAINT, CODE, SOUND un DATA, šādā secībā. Karte atrodas PAINT iekšpusē. SFX un dziesmas atrodas SOUND iekšpusē. Noklikšķiniet uz paneļa, lai tur strādātu. Kad CODE ir fokusā, SOUND un DATA ignorē burtu taustiņus, tāpēc rakstīšanas laikā netiek spēlēts arī klavieres.




Pilns izkārtojums, rakstīts `1:1:4:1:1` . Katrs skaitlis norāda, cik lielu daļu no rindas aizņem šis panelis: GAME, PAINT, CODE, SOUND, DATA. Mazs ekrāns atrodas centrētā 128 × 128 šūnā. Ekrāns, kas lielāks par 128 pikseļiem,

> paplašina GAME kolonnu.

IESTATĪJUMI → GLOBĀLIE → IZKĀRTOJUMS (YOUR LAYOUT) cikliski maina izkārtojumus:

| Kods | Ko jūs redzat |
|---|---|
| `1:1:4:1:1` | Viss. SPĒLE un SKAŅA sakrautas, KRĀSA un DATI sakrautas, KODS labajā pusē. |
| `1:1:4:1:1 2X` | Tas pats izkārtojums. KODS tiek zīmēts ar divreiz lielāku blīvumu, tāpēc tajā ietilps divreiz vairāk teksta. |
| `1:0:1:0:1` | SPĒLE, DATI, KODS. |
| `1:0:1:0:1 2X` | Tas pats izkārtojums, ar blīvāku KODU. |
| `0:1:1:1:1` | Nav spēles priekšskatījuma. KRĀSOŠANA, KODS, SKAŅA, DATI. |
| `0:1:1:1:1 2X` | Tas pats izkārtojums, ar blīvāku KODU. |
| `0:1:4:1:2` | Studija. Plats CODE, augstāka DATA lente, PAINT un SOUND. |
| `0:1:4:1:2 2X` | Tas pats izkārtojums, ar blīvāku KODU. |
| `0:0:4:0:4` | Tikai KODS un DATI, abi lieli. |
| `0:0:4:0:4 2X` | Tas pats izkārtojums, ar blīvāku KODU. |
| `0:1:4:1:0` | KRĀSA, SKAŅA un plašs KODS. |
| `0:1:4:1:0 2X` | Tas pats izkārtojums, ar blīvāku KODU. |
| `0:4:0:0:0` | Tikai KRĀSOŠANA, 256 × 256, viens lapas pikselis pret vienu audekla pikseli. |

`2X` rindā paneļi ir vienādi. CODE panelis tiek zīmēts ar divreiz lielāku izšķirtspēju un samazināts līdz tai pašai šūnai, tāpēc teksts, tabulītes un iedobes ir uz pusi garākas un ekrānā ietilpst divreiz vairāk avota teksta.

## SPĒLE

Darbojošās spēles kasetnes priekšskatījums. Noklikšķiniet uz paneļa SPĒLE vai nospiediet taustiņu kombināciju Ctrl+R, lai palaistu priekšskatījumu. IESTATĪJUMI → SPĒLES PANELIS ir vieta, kur tiek konfigurēts panelis:

- **Priekšskatījums** — krāsains, nepārtraukts vai automātiska atkārtota ielāde rediģēšanas laikā.

- **Spēlēt jaunā logā** — spēle atveras blakus redaktoram.
- **Displejs** — priekšskatījums kā RGB888 vai RGB565.
- **GAME 64** — 64×64 ekrāns ir vai nu centrēts šūnā proporcijā 1:1, vai arī mērogots 2×, lai to aizpildītu.
- **Ēnotāji** — CRT un filtri spēlē, atsevišķi no filtriem redaktorā.
- **Krāsa** — paneļa fons.

## KRĀSA

Spraitu lapas un tā paša paneļa kartes pusē — mozaīkas kartes. Lapas var būt lielākas par ekrānu. Spraitu priekšskatījuma izmēri ir 8, 16, 32, 64 un 128 (128 tikai tad, ja lapas izmērs ir vismaz 128 pikseļi). Ritiniet peles ritenīti virs спрайту priekšskatījuma, lai to tuvinātu. Otas izmēri ir 1, 2, 3 un 4.

| Atslēga | Rīks |
|---|---|
| B | Zīmulis |
| M | Taisnstūra atlase |
| Q | Laso |
| S | Pastmarka |
| G | Spainis |
| Es | Pipete |
| V | Pārvietot |
| Kreisais/Shift+L | Līnija/līkne |
| U/Shift+U | Tukšs taisnstūris / tukša elipse |
| F / Shift+F | Aizpildīts taisnstūris / aizpildīta elipse |
| D | Dīters |
| [ ] | Mazāka/lielāka birste |
| Ctrl+Z / Ctrl+Y | Atsaukt/aizvērt. Arī Ctrl+Shift+Z atceļ darbību. |
| Ctrl+C / Ctrl+X / Ctrl+V | Kopēt, izgriezt, ielīmēt |
| Ctrl+A | Atlasiet audeklu |

Karte izmanto to pašu ideju, ar mazāk rīkiem: B — zīmulis, M — taisnstūra atlase, S — zīmogs, V — pārvietošana un Ctrl+Z, Ctrl+Y, Ctrl+C, Ctrl+V. Turiet nospiestu Alt, lai tuvinātu kartes tekstūru. Otas izmērs kartē paliek nemainīgs, tāpat kā krāsošanas rīki.

Katram спрайтам ir astoņi karodziņu biti. Paint tos iestata uz спрайта; `fget` un `fset` tos nolasa spēlē.

IESTATĪJUMI → KRĀSAS REDAKTORS ir vieta, kur konfigurēt šo paneli:

- **Palete** — kura banka tiek izvilkta, kā arī ielādējiet un saglabājiet.
- **Priekšskatījums** — RGB888 vai RGB565.
- **Otas kursors** — rādītājs, ko izmantojat zīmēšanas laikā.
- **Pikseļu fiksācija** — kā ota piefiksē pikseļus.
- **Liela ota** — liela ota nokrīt no augšējā kreisā stūra vai no visa tās nospieduma.
- **Sīpolu miza** — citu kadru spoks un kurā virzienā šis spoks ir vērsts.
- **Kartes tālummaiņa** — īsti pikseļi jeb brīvā tālummaiņa.
- **Krāsas** — lietotāja saskarnes krāsas.

## KODS

Cilnes ir atsevišķi faili projekta ietvaros. Noklikšķiniet uz cilnes nosaukuma, lai to pārdēvētu; Enter apstiprina. Vairākas cilnes, kas katra definē `_init` , `_update` vai `_draw` ir apvienotas tabulēšanas secībā: pirmā cilne tiek palaista vispirms, pēdējā cilne tiek palaista pēdējā. Kods, kas neatrodas āķī, kļūst par `_init` daļu. Spēļu kasetne bez āķiem tiek uzskatīta par vienu `_draw` .

| Atslēga | Darbība |
|---|---|
| Bultas | Pārvietot kursoru. Shift paplašina atlasi. |
| Ctrl+kreisais/labais bulttaustiņš | Pēc vārda. |
| Sākums / Beigas, Ctrl+Sākums / Beigas | Rindas sākums vai beigas, vai tabulēšanas zīme. |
| Cilne | Ievietojiet divas atstarpes. Pabeigšanas sarakstā akceptējiet nosaukumu. |
| Ieiet | Jauna rinda, atkāpta tāpat kā iepriekšējā. Pieņem pabeigšanu. |
| Ctrl+C/X/V | Kopēt, izgriezt, ielīmēt. |
| Ctrl+Z / Ctrl+Y | Atsaukt/atkārtot. |
| Ctrl+A | Atlasiet cilni. |
| Ctrl+D | Dublēt līniju vai atlasi uz leju. |
| Ctrl+/ | Komentēt vai noņemt komentāru ar `--` . |
| Ctrl+F | Atrast. F3 pāriet uz nākamo rezultātu. Escape aizver joslu. |
| Ctrl+G | Dodieties uz rindas numuru. |

| | |
|---|---|
| Bēgšana | Vispirms aizver meklēšanas, došanās uz vai padomu sarakstu. Otrais Escape taustiņš atver redaktoru un atver izstrādātāja izvēlni. |

Rakstot funkcijas nosaukumu, tiek parādīti padomi. Nosaukumu ievieto taustiņš Enter vai Tab, un simbolu `(` , ja iestatījumos ir ieslēgta opcija COMPLETE (). IESTATĪJUMI → KODA REDAKTORS:

- **Meklēšana** — Ctrl+F aktīvajā cilnē vai visās cilnēs.
- **Augsts teksta augstums** — apgriezt glifus līdz 5 pikseļiem vai parādīt pilnu glifu.
- **Virknes lielums** — virknes literāļu fonta lielums.
- **Komentāru fonts** — 3×5 jeb pašreizējās lietotāja saskarnes valodas Fusion fonts.
- **Pabeigts (** — vai pabeigšana veido iekavas.
- **Koda norādes** — ieslēgtas vai izslēgtas.
- **Padomi valoda** — izvēlnes valoda jeb angļu valoda.
- **Iespiedkaršu atstarpe** — papildu nobīde tukšās rindās.
- **Notekcaurule** — rindu numuri.
- **Krāsa** — fons, komentāri, tabulatori, skaitļi, operatori, atslēgvārdi, virknes, programmas funkcijas, jūsu funkcijas, mainīgie, iekavas, kursorsvira, atlase, kļūdas rinda, brīdinājums.

## SKAŅA

Panelis ir aprīkots ar divām cilnēm. SFX ir viens modelis: īss nošu virkne. SOUND ir dziesma: tā novieto šos modeļus astoņos kanālos. TIC ir cik tikšu gaida viens solis. Pulkstenis ir 120 tikšķi sekundē, un viens solis ir sešpadsmitās daļas nots, tātad 15 tikši ir 120 sitieni minūtē. Augstāks TIC ir lēnāks modelis. Divi modeļi saglabā ritmu, kad viens TIC dala otru: 15 un 30 dala, 15 un 8 ne.

Klavieres skan, kad ar peli tiek atlasīta nots. Baltie taustiņi ir ZXCVBNM (CDEFGAB). Melnie taustiņi ir SDGHJ (C# D# F# G# A#). Novietojiet kursoru virs jebkuras vērtības un ierakstiet skaitli vai arī novietojiet kursoru virs tās un ritiniet ritenīti. Toņa augstuma skaitļi kodā ir no 0 (C-0) līdz 63 (D#5).

Vienā modelī LOOP un LEN ir vieni un tie paši divi skaitļi. OUT virs 0 atkārtojas no IN līdz OUT, un nosaukums ir LOOP. OUT pie 0 un IN virs 0 ir LEN: tik daudz nošu tiek atskaņotas no sākuma, tad modelis beidzas, un nosaukums ir LEN. Abi pie 0 atskaņo katru soli.

Cik daudz sintezatora var rediģēt, ir atkarīgs no spēles kasetnei izvēlētā bitu dziļuma. IESTATĪJUMI → SFX REDAKTORS iestata šo dziļumu, vai nots automātiski pārslēdzas uz priekšu (izslēgts, nošu ķēde vai lauka ķēde), vai atstarpes taustiņš restartē atskaņošanu vai pārslēdz sākšanu un apturēšanu, kā arī redaktora krāsas.

### 16 bitu

Katram solim ir nots, skaļums, vilnis un efekts. Skaļums ir no 0 līdz 7. 0 ir klusums.

Viļņi:

- **0 trīsstūris**
- **1 sasvērts zāģis**
- **2 zāģi**
- **3 kvadrāti**
- **4 pulss** — šaurs pulss
- **5 orgānu**
- **6 troksnis**
- **7 fāzers**
- **8 PWM** — kvadrāts, kura platums mainās
- **9 pulss 25%**
- **10 sinuss**
- **11 metālisks troksnis**
- **12 FM zvans**
- **13 plūkšana**
- **14 superzāģis**
- **15 PWM (C64)**

Efekti uz vienas notis:

- **0 nav**
- **1 slīdējums** — augstums un skaļums šajā solī slīd uz nākamo noti
- **2 vibrato** — skaņas toņa svārstības
- **3 kritums** — tonis krītas līdz klusumam, parastais spēriens
- **4. pakāpeniska parādīšanās**
- **5 izbalēt**
- **6 arpedžo** — iziet cauri šai notij un nākamajām trim, pa vienai ik pēc 8 tikšiem. Pie TIC 8 vai zemāk, ik pēc 4 tikiem.
- **7 ātras arpedžo** — tās pašas četras notis ik pēc 4 tikšiem vai ik pēc 2, ja TIC ir 8 vai zemāk.

Raksta galvene virs piezīmēm:

- **TIC** — tikšķi solī, no 1 līdz 255. 15 ir 120 sitieni minūtē. Ierakstiet skaitli vai soli pa solim mainiet to par vienu.
- **LOOP IN / OUT** — ja OUT ir lielāks par 0, raksts atkārtojas no IN uz OUT. Atstājiet OUT uz 0, un nosaukums kļūs par LEN: IN norāda atskaņoto nošu skaitu, un 0 nozīmē visu rakstu.
- **TROKSNIS** — viss raksts kļūst par troksni.
- **CRUSH** — katra viļņa rosīga versija.
- **DTUNE** — 0 izslēgts, 1 viegls piedziedājums, 2 spēcīgs piedziedājums.
- **RVERB** — atbalss: 0 izslēgta, 1 īsa, 2 gara.
- **DAMPN** — zemfrekvences filtrs: 0 izslēgts, 1 viegls, 2 spēcīgs.
- **LĪMENIS** — visa raksta skaļums, no 0 līdz 7.
- **ŠŪPOLES** — šūpošanās. Nedaudzi soļi gaida nedaudz ilgāk, vienmērīgi soļi nedaudz mazāk.
- **SOĻI** — cik soļu izmanto raksts: 32, 24, 16 vai 12.

### 8 bitu

Solim ir nots un vilnis. Četri viļņi ir kvadrāts, trīsstūris, zāģis un troksnis. Galvenē ir TIC, LOOP IN/OUT, FX un LEVEL. TIC ir viens no 1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 16, 20, 24, 30, 40, 60. FX ir viens efekts visam modelim: 0 nav, 1 vibrato, 2 arpedžo, 3 izbalēšana. LEVEL ir no 0 līdz 3.

### 4 bitu

Solis ir tikai nots. Galvenē ir TIC, LOOP (ieslēgts vai izslēgts), WAVE (tie paši četri viļņi) un OCT (oktāva, kurā atrodas notis). TIC ir viens no 4, 6, 8, 10, 12, 15, 20, 30.

### 2 bitu

Solis ir tukšs, bungas, sitiens vai saspiešana. Galvenē ir TIC un BASE. TIC ir 8, 15, 16 vai 30. BASE nosaka, cik augstu atrodas bungas un sitiens.

### 1 bits

Solis ir nots. TIC ir 8, 15, 16 vai 30 — tie paši četri ātrumi kā 2 bitu gadījumā.

### Dziesmas

Cilne SOUND (SKAŅA) ir atvērtā dziesma. Astoņas kolonnas ir kanāli. Šūna ir raksta ID. 255, tukša, nozīmē, ka kanāls šajā rindā ir kluss. X kolonna ir komanda: turpināt, apturēt, cikla sākums vai cikla beigas. Apturēt beidz dziesmu pēc rindas. Cikla beigas atgriežas pie pēdējā cikla sākuma.

Rinda beidzas, kad beidzas kreisajā malējais raksts ar LOOP OUT uz 0. Raksti tālāk pa labi šo rindu nepagarina: garāks tiek nogriezts, īsāks sākas no jauna no pirmā soļa, līdz rinda beidzas. Raksts ar LOOP OUT virs 0 veido cilpu starp LOOP IN un LOOP OUT un nebeidz rindu. Ja visi raksti rindā veido cilpu, rinda paliek tur.

### Īsceļi

| Atslēga | Darbība |
|---|---|
| Klavieru taustiņi | Uzrakstiet piezīmi par ar peli atlasīto darbību. |
| Augšup/Lejup | Mainiet vērtību zem kursora. |
| Kosmoss | Atskaņojiet atvērto skaņas efektu (SFX) vai atvērto dziesmu no pirmās rindas. Sāciet/apturējiet vai restartējiet SFX redaktora iestatījumos. |
| Ctrl+Z / Ctrl+Y | Atsaukt/atkārtot. |
| Ctrl+C / Ctrl+X / Ctrl+V | Kopēt, izgriezt, ielīmēt atlasi. |

## DATI

Nosauktas tabulas. Katra tabula sastāv no rindām un kolonnām. No koda šīs tabulas izmanto `dget` , `dset` , `dstr` , `drows` un `dcols` . Rindu un kolonnu numerācija kodā sākas ar 0.

Kolonnu tipi, cikliski parādīti galvenē:

- **bool** — viens bits.
- **2** — divi biti.
- **4** — četri biti.
- **8** — baits.
- **16** — 16 biti, id un lielākiem skaitļiem.
- **spr** — спрайта ID.
- **str** — teksts, kas zīmēts ar sistēmas fontu.
- **8.** un **10. raksts** — viens un tas pats teksts, tikai uzzīmēts garāk.
- **f16** un **f32** — pusprecīzs peldošais skaitlis un pilns peldošais skaitlis.

Kad rādītājs atrodas uz kolonnas galvenes, taustiņu kombinācija Ctrl+ritināšana maina šīs kolonnas veidu. Vienkārša ritināšana pārvietojas pa kolonnām vai rindām uz leju atkarībā no tā, ko atlasa IESTATĪJUMI → DATU REDAKTORS. Ctrl maina šo virzienu, tāpēc, nospiežot taustiņu Ctrl+ritināšana uz galvenes, mainās veids, kā ritenītis ir iestatīts uz kolonnām.

Bulttaustiņi pārvieto. Enter rediģē šūnu. IESTATĪJUMI → DATU REDAKTORS:

- **Ritenis** — ritiniet kolonnas vai ritiniet rindas.
- **Krāsas** — režģis, galvene un atlase.

## Aktīvi

Funkcija ASSETS importē un eksportē atvērto daļu, nevis visu projektu. Spraiti izmanto programmā PAINT atvērto спрайту uz atvērtās lapas. Karte ir atvērtā karte. SFX ir atvērtais raksts. Mūzika ir atvērtā dziesma. Dati ir atvērtā tabula. Izņēmums ir kods: katra cilne tiek uzrakstīta, un katra cilne tiek nolasīta atpakaļ.

| Laipns | Fails | Kas kustas |
|---|---|---|
| Spraiti | `.png` | Atvērtais spraits. |
| Karte | `.p37map` | Atvērtā karte. |
| Skaņas efekti | `.p37sfx` | Atvērtais raksts. |
| Mūzika | `.p37mus` | Atvērtā dziesma. Rakstu numuri paliek nemainīgi. Eksportējiet skaņas efektus atsevišķi, ja šīs skaņas ir nepieciešamas. |

| | | |
|---|---|---|
| Dati | `.p37dat` | Atvērtais galds. |
| Kods | `.lua` | Katra cilne. |

## Iestatījumi

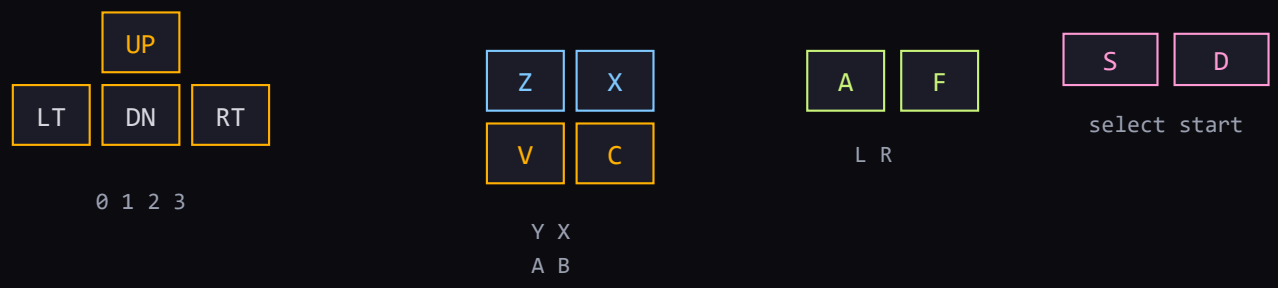
Sakņu saraksts:

| Lapa | Ko tas uzglabā |
|---|---|
| PROFILI | Saglabāt un ielādēt katru iestatījumu. Izveidot jaunu, pārdēvēt, pārrakstīt, dzēst. Kopēt vai ielīmēt profilu kā tekstu. |
| Izvēlne | Logs, taustiņi, izvēļņu krāsas, papildu fons. |
| SPĒĻU PANELIS | Priekšskatījums un atskaņošanas logs. |
| KODA REDAKTORS | Izcelt un norādīt. |
| KRĀSU REDAKTORS | Ota, palete, kartes skats. |
| SFX REDAKTORS | Sintezēšanas un nošu ievades funkcija. |
| DATU REDAKTORS | Riteņu virziens un krāsas. |
| VALODA | Izvēlnes un mājienu valoda. |
| LICENCES | Licences teksti, tostarp fonts Fusion Pixel un SDL3. |

GLOBĀLI:

- **IZKĀRTOJUMS** — paneļu izkārtojums, iepriekš.
- **SKAĻUMS** — meistara līmenis, no 0 līdz 10.
- **TAUSTIŅU SAISTĪŠANA** — taustiņi katram no 8 spēlētājiem. Nospiediet taustiņu, lai to notvertu. Norādījumi, ABXYLR, atlasīt.
- **Present** — izlaist dīkstāvē esošos kadrus vai uzzīmēt katru kadru.
- **Pilnekrāna režīms** — bez apmalēm.
- **Atsvaidzināt** — loga kadru skaits sekundē. Solis pa kreisi un pa labi: 60, 90, 120 un maks. Ievadiet skaitli līdz 1000. 0 nozīmē, ka nav ierobežojuma.
- **Vsync** — jāgaida attēlošana starp kadriem.
- **Mērogs** — vesela skaitļa loga reizinātājs.
- **Ēnotāji** — CRT un filtri redaktorā, nevis spēlē.

- **Krāsa** — izvēlnes teksts un iezīmējums.
- **Fons** — papildu PNG attēls, apgriezts līdz 128 pikseļiem, aiz izvēlnes. Notīrot to var noņemt.
- **Redaktora kursors** — rādītāja forma.
- **Pele** — peldošs rādītājs vai piesprausta pikseļos.
- **Lietotāja saskarnes ātrums** — cik ātri saglabātais virziens atkārtojas sarakstā.
- **Explorer** — failu saraksti sākas datora saknes direktorijā vai exe faila mapē.




Noklusējuma tastatūra. A ir apstiprināšanas poga izvēlnēs. B ir atgriešanās poga. Spēlē šīs ir 4. un 5. Izvēlēties (S) ir 10. poga, nevis izvēlnes saīsne.

| Poga | Loma | Noklusējuma atslēga |
|---|---|---|
| 0 | Kreisā | Kreisā bultiņa |
| 1 | Pa labi | Labi bultiņa |
| 2 | Augšup | Augšupvērsta bultiņa |
| 3 | Lejā | Lejupvērsta bultiņa |
| 4 | A, apstipriniet | V |
| 5 | B, mugura | C |
| 6 | X | X |
| 7 | Jā | Z |
| 8 | R | F |
| 9 | L | A |
| 10 | Atlasīt | S |
| 11 | Sākt | D |

## Būvēt

Izvēlieties platformu: Windows, Linux, Mac, Android, iOS vai Web.

- **Nosaukums** — būvējuma faila nosaukums.
- **Autors** — operētājsistēmās Windows un Linux. Šis ir saglabāto failu mapes nosaukums.
- **Steam lietotnes ID** — operētājsistēmās Windows un Linux. Tukšs nozīmē, ka nav Steamworks. Aizpildīts ID sasaista `achieve(0)` … `achieve(15)` ar Steam sasniegumiem ACH_0…ACH_15.
- **Id** — pakotnes identifikators, ja platforma to pieprasa.
- **Slēpt kodu** — darbvirsmas un mobilo ierīču pārklājumos. Tas palēnina spēles kasetnes `peek` . Tā nav slēdzene.
- **Pilnekrāna režīms** — eksportētais atskaņotājs tiek palaists pilnekrāna režīmā.
- **Ēnotāji** — līdz trim slāņiem, kas iestrādāti atskaņotājā.
- **Lejupielādēt .p37** — tikai tīmeklī. Pauzes izvēlnē tiek rādīts lejupielādes ieraksts.
- **Ikona** — flīze no спрайта lapas: lapa, izmērs, x, y. Vai arī PNG vāka attēls, ko apgriežat.
- **Eksportēt** — izvēlieties mapi. Izvēlnē tiek parādīts ziņojums BUILD SAVED (Izveidošana saglabāta) vai BUILD FAILED (Izveidošana neizdevās).

Izveidojiet būvējumu ar ieslēgtu internetu. Ievads var anonīmi jautāt serverim, vai šīs būvējuma hešs ir zināms. Šī pārbaude ir divu iemeslu dēļ. Tā aizsargā mani, PIX37 autoru, no rīka kopijām, kas nebija manas, lai tās nodotu tālāk. Tā arī aizsargā personu, kas izveidoja spēli. Kāda cita spēles ņemšana un tās norādīšana kā savas kļūst izplatīta tālruņu tirgū, piemēram, Play (Android) vai iOS. Spēle joprojām sākas. Ja ierīce ir bezsaistē, ievadā teikts, ka spēle nav savienota. Šī rinda ir arī mājiens spēlētājam, ka kopija, iespējams, nav nākusi no likumīga avota. Tā ir neliela pārbaude, lai, ja jūsu spēli kāds cits nozaga un pārdeva bez jūsu piekrišanas, jūs varētu iesniegt šo prasību veikalā, piemēram, Play (Android), Steam vai itch.io. Serveris blakus hešam saglabā laiku, kad hešs tika pirmo reizi reģistrēts, un vēlāk to nemaina. Agrākais laiks ir sākotnējā būvējuma laiks. Pēc tam reģistrēts hešs ir vēlāka kopija.

Es vēlos, lai šī pārbaude aizsargātu cilvēkus, kas veido spēles. Dzinējs jau tā paveic lielu daļu darba viņu vietā, un šī ir viena no šīm lietām. Es saprotu, ka tas var būt pretrunīgi. Ja kopiena nolems, ka labāk ir atteikties no šīs mikro-DRM tehnoloģijas, es to darīšu.

Veca spēles kasetne no iepriekšējā redaktora var atteikties ielādēties pēc valodas maiņas. Atveriet `.pix37` projektu šajā redaktorā un eksportējiet jaunu `.p37` failu.

## Izveidot savienojumu ar PIX37.PARTY

SAVIENOJUMS AR PIX37.PARTY ir lietotnē. Pārlūkprogrammā nav pieteikšanās lapas. Jūs piesakāties ar Steam sesijas biļeti vai itch.io, izmantojot planšetdatoru un peli. Pakalpojumā tiek parādīts unikāls nosaukums no 2 līdz 16 rakstzīmēm. Licences atslēga un Steam ID netiek rādīti citiem spēlētājiem.

Pēc pakalpojuma noteikumiem centrmezgls ir Izlase, Katalogs, Rangs, Spēlētāji un jūsu profils.

Izstrādātāja licence ir redaktors. Jūs izveidojat spēli, augšupielādējat to un atstājat zvaigznes. Spēlētāja licence ir spēlētājs. Sirds ir no cilvēkiem, kas spēlē skrējēju.

- **Katalogs** — pārlūkojiet spēles. Filtri aptver žanru (līdz trim), ekrāna izmēru, spēles kasetnes izmēru un to, vai augšupielādē ir iekļauts `.pix37` projekts.
- **Izlase** — spēles, kuras esat atzīmējis.
- **Augšupielāde** — nepieciešama izstrādātāja licence. Jūs nosūtāt `.p37` failu un varat pievienot `.pix37` projektu. `.p37` fails ir iepakots baitkods. To ir grūti lasīt, un to var apzināti maskēt. `.pix37` fails ir projekta fails, ar kuru jūs faktiski strādājat: avots, attēli un skaņa formātā, ko cita persona var atvērt un no kuras mācīties. Tā nosūtīšana ļauj citiem veidotājiem pētīt spēli, nevis saspiestu spēles kaseti. Kopā ar to ir norādīts nosaukums, autors, žanri, vāks, ikona un sasniegumu nosaukumi. Nosūtītais RAM numurs ir maksimālais apjoms, ko sasniedza spēles kasetne.
- **Komentāri un sasniegumi** — spēles lapā. Sasnieguma atbloķēšana, kamēr esat pieteicies, tiek saglabāta gan šajā kontā, gan ierīcē.
- **Rangs** — izstrādātāji, sakārtoti pēc zvaigznēm vai sirsniņām. Katrā rindā ir redzami abi skaitļi. Atveriet vārdu, lai skatītu šī autora spēles, sakārtotas pēc sirsniņām.
- **Spēlētāji** — konti, kas sakārtoti pēc atbloķētajiem sasniegumiem.

# Kods

Valoda ir neliels Lua stila skripts. Šī puse sākas ar tukšu projektu, māca valodu un pēc tam parāda īsu katras funkcijas piemēru.

## Pirmā programma

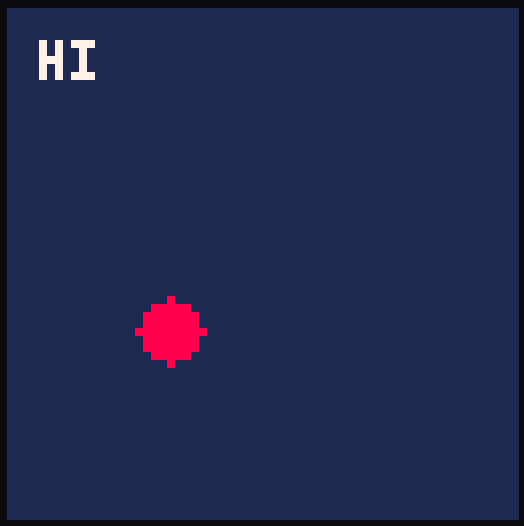
Izveidojiet projektu ar izmēru 64 × 64, 4 bitu krāsām. Atveriet CODE. Aizvietojiet tabulēšanas joslu ar:

```
_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 to palaiž. Bulttaustiņi pārvieto apli. Escape aptur spēli un atgriežas redaktorā.




Tie paši trīs izsaukumi: `cls(1)` uzzīmē tumši zilo ekrānu, `print` uzzīmē HI un `circfill` uzzīmē rozā apli.

## Kas darbojas un kad

- `_init` darbojas vienreiz.
- `_update` darbojas ar 60 Hz frekvenci. Šeit ievietojiet kustību un pogas.
- `_draw` zīmē. Bez fps argumenta tas zīmē tik ātri, cik vien dators un monitors spēj. `_draw(30)` aptur zīmēšanu pie 30 fps un atstāj loģiku pie 60.
- `_set_fps(n)` vēlāk maina šo ierobežojumu. 0 to notīra.

Nosaukumi nav reģistrjutīgi; redaktors tos saglabā ar mazajiem burtiem. `function` un `fun` ir viens un tas pats atslēgvārds.

## Valoda

### Programmēšanas pamati

Programma ir rindiņu saraksts, ko izpilda spēles kasetne. Katrs virsraksts vispirms parāda līnijas formu, pēc tam nelielu piemēru. Rinda, kas sākas ar `--` , ir komentārs: piezīme jums. Spēle to izlaiž.

### Mainīgie lielumi

Mainīgais ir nosaukums, kurā ir norādīta vērtība. Jūs izvēlaties nosaukumu. Nākamās rindas to var nolasīt un saglabāt jaunu vērtību ar to pašu nosaukumu.

Forma ir `name = value` . `local name = value` saglabā šo nosaukumu pašreizējā blokā, tāpēc pārējā spēles kārtridžs to nevar redzēt. Bez `local` nosaukums ir globāls, un to var redzēt katra cilne.

```
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` nozīmē, ka nav vērtības. `true` un `false` ir jā un nē. Nosaukums, ko nekad neesat iestatījis, lasāms kā `nil` .

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

### Galdi

Tabulā ir daudz vērtību ar vienu nosaukumu. Sarakstā tiek izmantoti skaitļi. Pirmais elements ir 1, nevis 0. Ierakstā tiek izmantoti nosaukumi. Tabulā var būt cita tabula.

Saraksta forma: `{ value, value }` . Ieraksta forma: `{ name = value, name = value }` . Ierakstu saraksts ir tabula, kuras elementi ir tabulas.

```
-- 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
```

### Stāvoklis un ķermenis

Nosacījums ir jautājums ar atbildi "jā" vai "nē", piemēram, `hp > 0` vai `btnd(4)` . Pamatdaļa ir rindu grupa, kas tiek izpildīta, ieejot blokā. `if` un katrs cikls izmanto vienu un to pašu formu: nosacījums, tad pamatdaļa un tad `end` . `repeat` noslēdzas ar `until` nevis `end` .

### Ja

`if` izpilda savu pamatfunkciju tikai tad, ja nosacījums ir patiess. `elseif` mēģina citu nosacījumu. `else` izpilda, ja visi nosacījumi ir nepatiesi.

Forma: `if condition then` , tad ķermenis, tad `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
```

### Cilpas

Cikls atkārto darbības pamattekstu. `while` vispirms pārbauda nosacījumu un turpina, kamēr tas paliek patiess. `repeat` palaiž darbības pamattekstu, pēc tam pārbauda, lai darbības pamatteksts vienmēr tiktu palaists vismaz vienu reizi. `for` . Pēdējais skaitlis tiek iekļauts. Bez trešā skaitļa solis ir 1, tāpēc, ja sākums jau ir aiz beigām, darbības pamatteksts netiek palaists. Atpakaļskaitīšanai ir nepieciešams negatīvs solis.

```
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
```

### Staigāšana pie galda

`all` apmeklē katru saraksta elementu, sākot ar pirmo. `pairs` apmeklē katru nosaukumu un vērtību, kas saglabāta zem šī nosaukuma.

```
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
```

### Funkcijas

Funkcija ir nosaukta recepte. Forma ir `function name(arguments)` , tad pamatteksts un tad `end` . To izsauc, ierakstot nosaukumu un iekavas. `return` nosūta atpakaļ vērtību, un tā var nosūtīt vairāk nekā vienu. Argumenti ir šīs funkcijas lokālie argumenti.

```
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` ir funkcija, kas var redzēt apkārtējos lokālos objektus. Tā ir slēgums. Tabulā var būt viena no tām, un to var izsaukt ar punktu.

```
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
```

### Jā, nē, un izvēle

`not` apgriež nosacījumu. `and` ir patiess tikai tad, ja abas puses ir patiesas. `a and b or c` `or b` , ja `a` ir patiess, pretējā gadījumā `c` . `cond ? a : b` ir tā pati ideja vienā rindā: `a` , kad nosacījums ir patiess, `b` , kad tas nav.

```
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)
```

### Matemātika un teksts

`+ - * /` ir saskaitīšana, atņemšana, reizināšana un dalīšana. `//` dala un saglabā veselu skaitli. `%` ir atlikums. `^` ir pakāpe. `..` savieno divus tekstus. `#t` ir saraksta elementu skaits. Salīdzinājumi ir `==` (vienāds), `~=` vai `!=` (atšķirīgs) un `< > <= >=` . `x += 1` pieskaita 1 pie x. Tā pati forma darbojas arī ar `-= *= /= ^= %=` .

```
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
```

### Kur dzīvo vietējais iedzīvotājs

`local` darbojas līdz `end` , kas aizver tā bloku. `if` / `end` ir tā paša bloks. Jūs varat pieminēt lokālo vērtību virs rindas, kas to iestata, bet līdz šīs rindas beigām vērtība ir `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
```

**Lēkšana**

`::label::` iezīmē rindu. `goto label` pāriet uz turieni, bet tikai tās pašas funkcijas ietvaros.

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

## Atmiņa, īsi sakot

`peek` un `poke` ļauj aplūkot tiešos pikseļus caur vienu 16 bitu adresi. Tie neglabā spēles kasetnes otru kopiju. Adreses `0x5F00` līdz `0x5FFF` ir reģistri: kamera, klips (tikai lasāms), zīmēšanas krāsa, aizpildījuma raksts, trīs palešu pārveidojumi un lappušu numuri. Baiti abās šī cauruma pusēs, 65280 no tiem, ir pikseļu logs.

Kad ekrāns ietilpst šajos 65280 baitos, 64×64 un 128×128 iekļaujas, bet 256×256 neietilpst, ekrāns sākas ar adresi 0, rinda pa rindai. Atlikusi ir спрайта lapa, kas atlasīta ar `0x5F41` (0 ir 0. lapa). Ja šī lapa ir lielāka par atlikušo vietu, `0x5F42` izvēlas lapu, un karte neatrodas logā. Ja lapa ietilpst, aste ir kartes šūnas no kartes, kas atlasītas ar `0x5F43` , pa vienam vai diviem baitiem katrā, un `0x5F44` sašķiro karti, ja tā neietilpst astē.

Kad ekrāns neietilpst, viss logs ir viena lapa ar 65 280 pikseļiem. `poke(0x5F40, n)` pāršķir ekrāna lapu. `poke(0x5F41, n)` ar n, kas nav 0, tā vietā rāda lapu n, un lapai 0 šajā režīmā nav skata. `poke(0x5F43, n)` ar n, kas nav 0, rāda karti n, un tas uzvar pār lapu. Karte 0 šajā režīmā ir tikai `mget` un `mset` .

`peek` , kas nenonāk uz pikseļa vai reģistra, atgriež 0. `poke` tur neko nedara. Šajā logā nav спрайта karodziņu, SFX un DATA tabulu. Izmantojiet `fget` , `sfxget` un `dget` .

## Praktisks pielietojums

Katra funkcija ir uzskaitīta šīs lapas apakšdaļā, kā arī praktiski PIX37 funkciju pielietojumi. Saites zem katra piemēra sniedz plašāku informāciju par funkcijām.

### Vienmērīga kustība

128×128 spēles kasetne, skats no augšas. Spraits 1 jau ir uzzīmēts. Bultas paātrinās jebkurā virzienā, un berze palēnina kustību. Spraits aptveras: puse, kas atstāj apakšu, tiek uzzīmēta augšpusē, un tas pats attiecas uz pārējām malām.

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



### Pārbauda, vai nākamais gājiens ir bloķēts

64×64 spēles kasetne. Grīda un labā siena jau ir kartē, uzzīmētas ar 3. spraitu, un šim спрайтам ir iestatīts 0. karogs. Bultiņas pārvieto 1. спрайтu. Vārds augšpusē norāda bloķēts, ja nākamais solis pārklātos ar nepārtrauktu flīzi, un brīvs, ja tas nepārklātos.

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



### Noslaucīšana ar apvēršanu

Nospiediet A. Noslaukšanas lauku zīmē flip(), pa vienai melnai rindai vienlaikus, nevis _draw() iekšpusē. _draw netraucē, kamēr tā darbojas, tāpēc katra rinda paliek ekrānā.

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



### Funkciju tabula (klase)

Spraits 1 jau ir uzzīmēts. make_monster darbojas kā klase. Tas atgriež tabulu, un update un draw ir metodes, kas saglabā šī monstra x un y. Skaitlis augšpusē norāda, cik tālu tas ir nogājis.

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



### Četri pogas nolasīšanas veidi

A ir poga 4. Turiet to nospiestu, un lādiņš pieaug. Atlaidiet, un lādiņš atgriežas pie 0. Turot nospiestu, atkārtojas kustības uz augšu un uz leju, tāpēc izvēlnes rinda turpina kustēties. btnd ir pirmais nospiešanas kadrs, ko šī spēles kasetne izmanto tikai kā nospiešanas sākumu.

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



### Šauj ar peli

Spraits Nr. 1 jau ir uzzīmēts vidū. Balts punkts ir kursors. Kreisais klikšķis raida pikseli tā virzienā. Ritenis maina ātrumu, kas ir attēlots stūrī.

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



### Ierakstiet vārdu un apstipriniet to

Nospiediet A, lai atvērtu lauku. Ar bultiņām uz augšu un uz leju var mainīt mirgojošu burtu. Ar A var ierakstīt, ar X dzēst, ar Y var atstarpi, un ar B var apstiprināt. Pēc tam rindā tiek parādīts jūsu vārds.

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



### Kamera uz atskaņotāja

128×128 spēles kasetne. Spraits 1 sākas flīžu grīdas vidū. Bultiņas pārvietojas. Kamera notur spraitu centrā. x un y stūrī neslīd, jo camera() bez argumentiem atgriež skatu HUD.

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



### Dažas vizuālā romāna rindiņas

Rindas ir DATA formātā, angļu un poļu valodā. A turpinās jeb izvēles paliek, kad ekrānā ir pieejama izvēle. B izvēlas izvēles aiziešanu, pretējā gadījumā maina valodu. X maina valodu, kamēr tiek rādīta izvēle. Dialoglodziņā tiek izmantots 8px fonts, lai zīmētos poļu burti.

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



### Gleznojiet ar peeping un bakstīt

64×64 spēļu kasetne, tātad adrese 0 ir augšējais kreisais pikselis, un nākamā rinda sākas ar 64. Spēles kasetne ieraksta pikseļus, kopē rindu, notīra rindu un izdrukā nolasītās vērtības.

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



### Saraksts, ko varat pievienot

A pieskaita skaitli līdz 8. B noņem pēdējo. X noņem pirmo. Saraksts tiek izdrukāts pa ekrānu.

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



### Vārds, kas izjaukts

Spēļu kārtridžs sadala ada,12,blue un izdrukā nosaukumu, tā pirmo burtu — 12 — un vārdu blue.

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



### Atskaņojiet noti un pēc tam mainiet to

0. modelis ir vienas nots garš, un katrs atkārtojums ir ar vienādu skaļumu. A to atskaņo vēlreiz, kad šī nots ir beigusies. B paaugstina skaņas augstumu un atskaņo šo noti.

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



### Dziesma, kas ierakstīta spēles kasetnē

0. rindā atkārtojas divi ritmi, pa vienam katrā no pirmajiem diviem kanāliem. Rinda paliek 0. ch0 un ch1 ir nots, uz kuras atrodas katrs kanāls. Rindu LEN un LOOP rediģēšana spēles kasetnē ir veids, kā redzēt, ko šīs vadības ierīces dara mūzikas panelī.

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



### Trīs monstri DATOS

DATA panelī jau ir nosaukuma kolonna un dzīvspēka kolonna ar gļotām, sikspārni un aci. A bojā gļotas. Skaitļi labajā pusē ir tā dzīvspēks.

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



### Atcerieties rezultātu

A pievieno punktu un saglabā to. Iziet no spēles un atver spēles kasetni vēlreiz: skaitlis joprojām ir tur. B to aizmirst.

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





### Otra nodaļa citā failā

Divas spēļu kasetnes koplieto saglabāšanas slota uzdevumu. 1. daļa: A apzīmē spēles pabeigšanu. 2. daļa, kas tiek atvērta pirmā, norāda pabeigt 1. daļu. Pēc tam tiek atvērta 2. daļa, un B izdzēš saglabāto ierakstu, lai 1. daļu varētu spēlēt vēlreiz.

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



### Viens sasniegums

8. sprite jau ir izvilkts. Nospiediet taustiņu A desmit reizes. Uzraksts nosauc sasniegumu, un 8. sprite paliek, jo tas ir atbloķēts. Taustiņš B notīra šo sasniegumu lokālajā failā un atiestata skaitītāju.

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



### Istaba, kurā var ieiet

256×256 spēles kasetne. Balts punkts ir kursors. Izvēlne ir host, join vai serveru saraksts. Host atver istabu ar nosaukumu walk un sāk spēli. Join ritina adresi (cipari, punkts, kols), un B izveido savienojumu. Serveru saraksts pārvietojas ar augšup un lejup vērstiem bultiņām, un A pievienojas rindai. Spēlē bultiņas staigā спрайтā 1. Viena monēta tiek dalīta: host to pārvieto, un, to paņemot, šī spēlētāja rezultātam tiek pievienots 1 un tiek izveidots nākamais. Katrs спрайts, ieskaitot jūsu, blakus rāda šo rezultātu. Stūrī ir parādīts, cik spēlētāju

ir, LAN adrese un host publiskā adrese. B aiziet: jūs un pārējie atgriežaties izvēlnē, un nākamais host sāk jaunu istabu ar rezultātu 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`



### Dialogs, kas gaida

A parāda nākamo rindu. Funkcija apstājas pie katras atzīmes, līdz vēlreiz nospiežat taustiņu A.

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



### Pagaidiet pussekundi

Pēc aptuveni pussekundes vārds mainās no ... uz hi. Kamēr korutīna ir apturēta, ekrāns turpina zīmēt.

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



### Apļi, kas ierodas viens pēc otra

No augšas pil pieci apļi, divu kadru attālumā viens no otra, un tad krīt.

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



### Divi kadri no pulksteņa

1. un 2. cпрайts ir divi viena un tā paša tēla kadri. Attēls mainās četras reizes sekundē.

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

## Katra funkcija

Argumenti kvadrātiekavās nav obligāti. Tabulu un virkņu indeksi sākas ar 1, izņemot DATA rindas un kolonnas, kas sākas ar 0. Krāsas ir paletes indeksi.

### Indekss

`_zīmēt`
`_init`
`_mp`
`_mp_fail`
`_mp_join`
`_mp_leave`
`_mp_update`
`_set_fps`
`_atjaunināt`
`abs`
`achclear`
`achdef`
`sasniegt`
`sasniegts`
`pievienot`
`visi`
`apgalvot`
`atan2`
`josla`
`bnot`
`bor`

btn
btnd
btnp
btnu
bxor
kameru
griestus
chr
circ
circfill
noklikšķiniet uz
clickm
klikšķinātājs
klips
kls
kopradīt
krāsa
coresume
cos
costatus
skaitīt
kursors
dcols
del
delikateses
delsave
dget
drows
dset
dstr
fget
fillp
uzsist
flr
foreach
fset
info
ipairs
atslēgu izslēgšana
keyon
keystat
keystr
līniju
slodze
žurnāls
lshr
karte
maks
memcpy
memset
mget
vidus
min
pele
mset
mūzika
musicstat
paziņot
zvanīt
nākamais
nhost
njoin
nkb
nkbstr
nlan
nleve
nlist
nklausies
piezīme
nplayers
nsend
nstun
ord
ovāls
ovāls aizpildījums
iepakojums
pāriem
draugs
palrgb
palt
pcall
palūrēt
palūrēt2
palūrēt4
perf
pget
bakstīt
kule2
kule4
drukāt
drukāt 10
drukāt8
pset
taisn
taisnais aizpildījums
rnd
rotl
rotr
pareizi
rrectfill
saglabāt
saglabāšanas vieta
ekrāns
ritiniet
atlasiet
setmap
komplektu palete
setspr
setex
uzstādījums
sfx
sfxget
sfxhdr
sfxset
sfxstat
sget
sgn
shl
šovu statistika
shr
grēks
dziesma
dziesmu kopa

| | | |
|---|---|---|
| sadalīt | sspr | tonum |
| spr | stat | tostr |
| sqrt | apakš | izpakot |
| srand | laiks | whoami |
| sset | tline | raža |

## Dzīves cikls

### _init

`_init() ... end`

Tiek palaists vienreiz, kad tiek startēta spēles kasetne. Kods, kas atrodas ārpus jebkura āķa, arī tiek ievietots _init.

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

### _update

`_update() ... end`

Spēles loģika. Izsaukta ar 60 Hz frekvenci, pat ja _draw ir ierobežots vai resursdators ir lēnāks. Lasiet pogas šeit.

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

### _draw

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

Uzzīmē vienu kadru. Bez argumenta vai ar 0 tas tiek palaists tik bieži, cik resursdators to parāda. Nododiet n (1..999), lai ierobežotu zīmēšanu ar n kadriem sekundē. _update paliek pie 60 Hz.

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

### _set_fps

`_set_fps(n)`

Maina _draw ierobežojumu spēles darbības laikā. 0 nozīmē neierobežotu vērtību. n tiek ierobežots līdz 0..999.

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

### flip

`flip()`

Parāda ekrānu un piekāpjas līdz nākamajam kadram. Izmantojiet to tikai tad, ja zīmējat savu kadru ciklu. Neaizņemieties — gaidiet pēc tam.

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

### time

`time()`

Sekundes kopš _init.

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

## Ievade

### btn

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

1, kamēr tiek turēta poga b, pretējā gadījumā 0. p ir spēlētājs, 0..7. Bez argumentiem tas atgriež 0. spēlētāja bitu masku. Pogas: 0 pa kreisi, 1 pa labi, 2 uz augšu, 3 uz leju, 4 A, 5 B, 6 X, 7 Y, 8 R, 9 L, 10 atlasīt, 11 sākt.

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

### btnd

`btnd(b, [p])`

Patiesa tikai _update kadrā, kad nospiesta poga b.

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

### btnp

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

Nospiežot pogu, tā ir patiesa, un atkal, turot nospiestu: 30 tikšķi ar aizkavi, pēc tam ik pēc 8 tikšķiem. Bez pogas atgriež masku.

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

### btnu

`btnu(b, [p])`

Tiesa, tikai uz rāmja poga tiek atlaista.

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

### mouse

`mouse()`

Atgriež peles x un y koordinātas spēles pikseļos.

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

### clickl

`clickl([p])`

Kreisā peles poga. clickl() tiek turēta nospiesta. clickl(1) ir nospiešanas mala.

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

### clickr

`clickr([p])`

Peles labā poga. Tā pati 1 argumenta mala kā klikšķim.

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

### clickm

`clickm([p])`

Vidējā poga vai klikšķis uz ritināšanas ritenīša.

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

### scroll

`scroll()`

Peles ritenīša kustība šajā rāmī. Pozitīvais un negatīvais spēks ir atkarīgs no ritenīša virziena.

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

### keyon

`keyon()`

Atver teksta lauku, ko vada pogas. Augšup un lejup vērstie bultiņas maina burtus no A līdz Z. A ieraksta šo burtu. X dzēš. Y ir atstarpe. B apstiprina un aizver lauku. Ar tastatūru tajā nevar rakstīt.

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

### keyoff

`keyoff()`

Paslēpj teksta lauku.

```
keyoff()
```

### keystr

`keystr([s])`

Teksts laukā, tad 1 rāmī B apstiprina, tad burts augšpusē un apakšā mainās. Šis burts ir tukšs, kad lauks tiek aizvērts. keystr("") notīra tekstu.

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

### keystat

`keystat()`

1, kamēr teksta lauks ir atvērts, pretējā gadījumā 0.

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

## Zīmējums

### screen

`screen()`

Pašreizējā spēles kasetnes ekrāna izmērs pikseļos. Atgriež platumu, pēc tam augstumu. Šī ir spēles izšķirtspēja, nevis redaktora logs.

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

### cls

`cls([col])`

Notīra ekrānu. Noklusējuma krāsa ir 0.

```
cls(1)
```

### color

`color(col)`

Iestata zīmēšanas krāsu, kas tiek izmantota, ja vēlākā izsaukumā netiek izmantota col. Bez argumenta tā atgriež pašreizējo krāsu.

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

### camera

`camera([x, y])`

Nobīda katru zīmēšanas izsaukumu. camera() atiestata uz 0, 0.

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

### clip

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

Ierobežo zīmēšanu līdz taisnstūrim. Piektais arguments, kas nav nulle, krustojas ar pašreizējo klipu. clip() atjauno pilnekrāna režīmu. Klipu reģistru ieskats ir tikai lasāms.

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

### cursor

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

Nosaka, kur nonāks nākamā izdruka bez koordinātām, un pēc izvēles arī krāsu. cursor() atgriež x, y.

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

### pset

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

Iestata vienu ekrāna pikseli. Tiek ieskaitīts pikseļu statistikā.

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

### pget

`pget(x, y)`

Ekrāna pikseļa krāsa x un y asīs.

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

### print

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

Zīmē 3×5 tekstu (UTF-8, apgriezts no Fusion fonta). Bez x un y tas tiek drukāts pie kursora un pāriet uz nākamo rindu.

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

### log

`log(...)`

Pievieno argumentus, atdalītus ar atstarpēm, teksta buferim. pix37_run izdrukā šo buferi standarta izejā. Tas nezīmē.

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

### print8

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

Tāpat kā drukātā veidā, ar 8 px Fusion fontu.

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

### print10

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

Tāpat kā drukātā veidā, ar 10 px Fusion fontu.

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

### line

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

Līnija no x0,y0 līdz x1,y1. No iepriekšējās rindas turpinās divi argumenti. line() atiestata šo ķēdi.

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

### circ

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

Apļa kontūra. Rādiuss ir pikseļos.

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

### circfill

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

Aizpildīts aplis.

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

### oval

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

Elipses kontūra ierobežojošā lodziņa iekšpusē. Stūri ir iekļauti.

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

### ovalfill

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

Aizpildīta elipse ierobežojošā lodziņā.

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

### rect

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

Taisnstūra kontūra. Stūri ir iekļauti.

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

### rectfill

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

Aizpildīts taisnstūris.

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

### rrect

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

Noapaļota taisnstūra kontūra. x un y ir augšējā kreisā stūra. w un h ir izmērs pikseļos; abiem jābūt lielākiem par 0. r ir stūra rādiuss un pēc noklusējuma ir 0. Tas ir piestiprināts pie puses no īsākās malas.

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

### rrectfill

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

Aizpildīts noapaļots taisnstūris. Tādi paši argumenti kā pareizajam.

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

### fillp

`fillp([p])`

16 bitu aizpildījuma raksts vēlāk aizpildītām formām. 0 ir nepārtraukta vērtība. fillp() atgriež pašreizējo rakstu. Biti iet pa 4×4 rakstu.

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

### pal

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

Pārveido krāsu c0 par c1. 0. režīms ir zīmēšanas palete, 1. režīms ir attēlošanas palete, 2. režīms ir aizpildījuma raksta palete. pal() atiestata.

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

### palrgb

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

Iestata vienu paletes ierakstu uz RGB, katru kanālu no 0..255.

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

### palt

`palt([col, t])`

Caurspīdīgums. palt(col) vai palt(col, true) padara col caurspīdīgu. palt(col, false) padara to vienkrāsainu. palt() padara visas krāsas necaurspīdīgas.

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

### setpalette

`setpalette(n)`

Atlasa palešu banku n. setpalette() atgriež pašreizējo banku.

```
setpalette(1)
```

### tline

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

Zīmē līniju ekrānā, vienlaikus ņemot kartes pikseļu paraugus no kartes pozīcijām mx, my, mdx, mdy, un pakāpeniski pārvieto kartes paraugu pa līniju.

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

## Spraiti un kartes

### spr

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

Zīmē спрайту n punktos x un y. w un h ir izmērs pikseļos, pēc noklusējuma ir 8. fx un fy apgriež horizontāli un vertikāli, ja vērtība nav nulle. tex ir спрайта lapas tabulēšanas leņķis. rot ir leņķis pagriezienos (0..1), tajā pašā virzienā kā sin un cos. 0 atstāj спрайту nerotētu.

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

### sspr

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

Kopē спрайtu lapas taisnstūri uz ekrānu, mērogā dw reiz dh. Noklusējuma iestatījumi saglabā avota izmēru.

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

### sget

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

Viena teksela krāsa uz спрайtu lapas.

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

### sset

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

Uzraksta vienu tekselu uz спрайtu lapas.

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

### fget

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

Spraitu karodziņi. fget(n) atgriež karodziņa baitu. fget(n, f) ir "true", ja ir iestatīts karodziņa bits f (0..7).

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

### fset

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

Iestata спрайtu karodziņus. fset(n, v) ieraksta visu baitu. fset(n, f, v) iestata vienu bitu.

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

### setspr

```
setspr(n)
```

Atlasa aktīvo спрайtu lapu. setspr() atgriež pašreizējo lapu. Tā pati banka kā settex.

```
setspr(1)
```

### settex

```
settex(n)
```

setspr pseidonīms.

```
settex(0)
```

### map

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

Zīmē karti. Kartes šūna cx, cy tiek uzzīmēta ekrānā sx, sy. cw, ch, ja 0 nozīmē visu karti. Slānis tiek pieņemts un ignorēts. tex ir mozaīkas kopas lapa.

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

### mget

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

Spraita ID kartes šūnā cx, cy.

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

### mset

`mset(cx, cy, n)`

Ieraksta спрайtu n kartes šūnā cx, cy.

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

### setmap

`setmap(n)`

Atlasa n. karti. setmap() atgriež pašreizējo karti. 0. karte ir reāla karte.

```
setmap(1)
```

## Atmiņa

### peek

`peek(addr)`

Nolasa vienu baitu no 64 KB loga. Logs ir tiešraides ekrāns, pēc tam veido спрайтus pikseļos un tad kartē šūnas, kad ekrāns ietilpst 65280 baitos. Lielāki ekrāni tiek pārlapoti ar poke(0x5F40, page). Adreses, kas nav kartētas, atgriež 0. Rakstīts arī kā @addr.

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

### poke

`poke(addr, ...)`

Raksta baitus uz adresi (addr). Papildu argumenti raksta šādus baitus. Neatzīmētās adreses tiek ignorētas. Raksta arī $addr, val. Ekrāna atvēršana tiek uzskatīta par pikseļa ierakstīšanu.

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

### peek2

`peek2(addr)`

Neparakstīta 16 bitu vērtība, mazā endiāna, adresē.

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

### poke2

`poke2(addr, val)`

Raksta neparakstītu 16 bitu vērtību, mazo endianu.

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

### peek4

`peek4(addr)`

Neparakstīta 32 bitu vērtība, mazais endiāns.

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

### poke4

`poke4(addr, val)`

Raksta neparakstītu 32 bitu vērtību, mazo endianu.

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

### memcpy

`memcpy(dst, src, len)`

Kopē len baitus tajā pašā 64 KB logā.

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

### memset

`memset(dest, val, len)`

Aizpilda len baitus, sākot no dest ar val.

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

## Matemātika

### abs

`abs(x)`

Absolūtā vērtība.

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

### flr

`flr(x)`

Noapaļo uz mīnus bezgalību.

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

### ceil

`ceil(x)`

Noapaļo uz plus bezgalību.

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

### max

`max(a, b)`

Lielākā no abām vērtībām.

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

### min

`min(a, b)`

Mazākā no abām vērtībām.

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

### mid

`mid(a, b, c)`

Vidējā no trim vērtībām. Noderīga kā skava.

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

### sgn

`sgn(x)`

X zīme: -1, 0 vai 1.

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

### sqrt

`sqrt(x)`

Kvadrātsakne. Negatīvs arguments nav skaitlis.

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

### sin

`sin(x)`

Sinuss. x ir pārmaiņus: 0, 0,25, 0,5, 0,75 apriņķo apli vienu reizi. Nevis radiānos.

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

### cos

`cos(x)`

Kosinuss, savukārt, ir tāds pats apzīmējums kā sin.

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

### atan2

`atan2(dx, dy)`

Vektora dx, dy leņķis pagriezienos no 0 līdz 1.

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

### rnd

`rnd([x])`

Nejaušs skaitlis [0, x]. Noklusējuma x ir 1, tāpēc rnd() ir 0..1. Metamam kaudzim, flr(rnd(6))+1.

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

### srand

`srand(seed)`

Ievada nejaušo vērtību ģeneratora sākuma vērtību, lai vēlāku apļu secību varētu atkārtot.

```
srand(1)
```

### band

`band(a, b)`

Bitu UN. Rakstīts arī kā a un b.

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

### bor

`bor(a, b)`

Bitu VAI. Rakstīts arī kā a | b.

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

### bxor

`bxor(a, b)`

Bitu XOR. Rakstīts arī kā ^^ b.

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

### bnot

`bnot(a)`

Bitu līmenī NAV. Rakstīts arī kā ~a.

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

### shl

`shl(a, n)`

Pabīdīt pa kreisi. Rakstīts arī kā << n.

### shr

`shr(a, n)`

Aritmētiskā nobīde pa labi. Rakstīts arī kā >> n.

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

### lshr

`lshr(a, n)`

Loģiska nobīde pa labi. Rakstīts arī kā >>> n.

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

### rotl

`rotl(a, n)`

Pagriezt bitus pa kreisi. Rakstīts arī ar <<> n.

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

### rotr

`rotr(a, n)`

Pagriezt bitus pa labi. Rakstīts arī >>< n.

## Tabulas un vērtības

### add

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

Pievieno v tabulai t. Ja ir dots i, ievieto šajā uz 1 balstītajā indeksā.

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

### del

`del(t, v)`

Noņem pirmo ierakstu, kas vienāds ar v.

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

### deli

`deli(t, [i])`

Noņem indeksu i un atgriež tā vērtību. Ja i nav, noņem pēdējo ierakstu.

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

### all

`all(t)`

Iterators pāri masīva daļai secībā. Caurumi aptur gājienu tāpat kā 1 bāzes masīvs beidzas.

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

### foreach

`foreach(t, fn)`

Izsauc fn(v) katram masīva elementam.

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

### pairs

`pairs(t)`

Iterators pār katru atslēgu un vērtību, ieskaitot nosauktos laukus.

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

### ipairs

`ipairs(t)`

Iterators masīva daļā ar indeksu 1.

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

### next

`next(t, [k])`

Atgriež nākamo atslēgu un vērtību pēc k. next(t) sākas no sākuma. Izmanto pāri.

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

### count

`count(t, [v])`

Masīva daļas garums. Ja norādīts v, saskaita, cik ierakstu ir vienādi ar v.

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

### pack

`pack(...)`

Iepako argumentus jaunā tabulā.

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

### unpack

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

Atgriež no t[i] līdz t[j] kā atsevišķas vērtības. Noklusējuma vērtības aptver visu masīva daļu.

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

### select

`select(i, ...)`

select("#", ...) atgriež, cik argumentu seko. select(i, ...) atgriež argumentus no i uz priekšu.

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

### assert

`assert(c, [msg])`

Apstājas ar kļūdu, ja c ir false vai nil. msg aizstāj noklusējuma tekstu.

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

### pcall

`pcall(f, ...)`

Izsauc funkciju f. Veiksmes gadījumā atgriež vērtību "true" un rezultātu. Kļūdas gadījumā atgriež vērtību "false" un kļūdas tekstu.

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

## Stīgas

### chr

`chr(n)`

Viena rakstzīme no baitkoda 0..255.

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

### ord

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

Rakstzīmju "s" baitkodi no indeksa "i" līdz "j". Indeksi ir balstīti uz 1. Noklusējuma vērtība ir pirmā rakstzīme.

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

### sub

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

Apakšvirkne no i līdz j, balstīta uz 1. Izlaidiet j, lai iegūtu asti.

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

### split

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

Sadala s tabulā. *sep* pēc noklusējuma ir komats. Skaitlis *sep* ir fragmenta platums. *conv* konvertē daļas ar *tonum*, ja tas ir *true*.

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

### tonum

`tonum(s, [base])`

Parsē skaitli. Bāze var būt 2..36. Atgriež nulli, ja teksts nav skaitlis.

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

### tostr

`tostr(v, [digits])`

Pārvērš vērtību tekstā. "digits" iestata decimālvietas skaitļiem.

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

## Audio

### sfx

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

Atskaņo rakstu n un atgriež kanālu. Izlaižot ch vai atstājot -1, tiek izvēlēts pirmais brīvais kanāls. ch -2 aptur rakstu n katrā kanālā. n -1 aptur kanālu vai visus kanālus, ja ch tiek izlaists. n -2 ļauj cikla rakstam pabeigties, nevis atkārtot ciklu. sfx() bez argumentiem atgriež -1.

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

### sfxget

`sfxget(n, step)`

Atgriež viena modeļa soļa tabulu: [1] augstums, [2] vilnis, [3] skaļums, [4] efekts. Augstums ir 0, 63 (no C-0 līdz D#5). Vilnis ir 0, 15. Skaļums ir 0, 7, un 0 ir kluss. Efekts ir 0, 7.

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

### sfxset

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

Uzraksta vienu šablona soli. Nepieciešami visi seši argumenti.

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

### sfxhdr

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

Raksta galvene. sfxhdr(n) atgriež galvenes tabulu. sfxhdr(n, key, val) iestata vienu lauku: speed, loop_in, loop_out, level, noise, crush, dtune, rverb, dampn, swing, steps. Skaitļi no 0 līdz 10 atlasa tos pašus laukus šādā secībā. speed ir TIC, tikšķi uz soli. Augstāks ir lēnāks. 15 ir 120 sitieni minūtē. 16 bitu versija pieņem 1 līdz 255. Šaurāki sintezatori piesaistās savam TIC sarakstam.

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

### sfxstat

`sfxstat([ch])`

Kanālā atskaņotā raksta ID ir ch vai -1. Noklusējuma kanāls ir 0.

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

### note

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

Atskaņo vienu noti bez modeļa. augstums 0..63, vilnis 0..15, skaļums 0..7, efekts 0..7. Tikšķi norāda, cik ilgi tas turas, tajos pašos 120 Hz tikšķos kā TIC. Izlaižot tikšķus, tas ilgst 15, viens noklusējuma solis. Efekts darbojas visā šajā laikā, pēc tam kanāls apklust.

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

### music

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

Sāk dziesmu n no augšas. music(-1) apstājas. Izsaucot music ar ID, kas jau tiek atskaņota, tā paliek darbojoties. music() atgriež atskaņotās dziesmas ID jeb -1. fade ir milisekundes: dziesmas skaņas līmenis samazinās vai samazinās, kad n ir -1. Iestatītie maskas biti ir dziesmai izmantotie kanāli. 0 nozīmē visus astoņus. Pārējie kanāli paliek brīvi sfx.

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

### musicstat

```
musicstat()
```

Atskaņojamās dziesmas tabula. “song” un “row” ir dziesmas ID un rinda vai -1, ja nekas netiek atskaņots. “step” un “pos” ir astoņi cipari. “step[1]” ir 0. kanāls, “step[8]” ir 7. kanāls. “step” ir nots indekss vai -1, ja kanāls ir kluss. “pos” norāda, cik tālu nots ir aizgājusi, no 0 līdz 1. “tick” norāda, cik tikšu šajā rindā jau ir atskaņota. “played” norāda, cik rindu ir sāktas kopš music() funkcijas, ieskaitot to, kas pašlaik tiek atskaņota.

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

### songget

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

Raksta ID tiek glabāts dziesmas šūnā. ch ir 0..7. 255 nozīmē, ka šūna ir tukša. Šis ir tiešsaistes izsaukums; koda padoma tekstā joprojām ir redzama vecāka argumentu secība.

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

### songset

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

Ieraksta dziesmas šūnā raksta ID. Izmantojiet 255, lai notīrītu šūnu.

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

### stat

`stat(n)`

Tikai audio. 16..23 ir skaņas efekta ID kanālā n-16. 24..31 ir mūzikas modelis kanālā n-24. Jebkurš cits n atgriež 0. Kadru laiks un RAM ir perf un info, nevis stat.

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

## Datu tabulas

### dget

`dget(t, row, col)`

Nolasa datu tabulas t šūnu no DATA paneļa. Rinda un kolonna sākas ar 0.

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

### dset

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

Raksta šūnu. Izmantojiet virkni teksta kolonnai.

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

### dstr

`dstr(t, row, col)`

Šūna kā teksts. Skaitļi tiek konvertēti.

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

### drows

`drows(t)`

Cik rindu ir tabulā t?

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

### dcols

`dcols(t)`

Cik kolonnu ir tabulā t?

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

## Saglabāšana un sasniegumi

### save

`save(k, v)`

Saglabā v zem atslēgas k saglabāšanas failā. v var būt skaitlis, virkne vai šo vērtību tabula. Fails tiek saspiests.

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

### load

`load(k)`

Nolasa atslēgu k. Atgriež nulli, ja atslēgas trūkst.

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

### delsave

`delsave(k)`

Izdzēš atslēgu k no saglabāšanas.

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

### saveslot

`saveslot([name])`

Izvēlas saglabāšanas failu autora mapē. Ja nosaukuma nav, slots ir spēles nosaukums. Atgriež 1, ja šis fails jau pastāv.

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

### achieve

`achieve(i)`

Atbloķē sasniegumu i no 0 līdz 15. Saglabāts lokāli. Steam versijā tiek iestatīts arī ACH_i. Kataloga sesija ziņo par to spēlētāja kontā.

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

### achieved

`achieved(i)`

1, kad i-tais sasniegums jau ir atbloķēts.

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

### achdef

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

Uzrunas teksts par sasniegumu 1, kas tiek rādīts 15 sekundes no ekrāna apakšas. SPR ir papildu 8x8 flīze, apskatiet tās lapu.

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

### achclear

`achclear(i)`

Notīra sasniegumu i lokālajā failā. Jau iestatīts Steam ACH_i paliek iestatīts.

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

## Tīkls

### whoami

`whoami()`

Jūsu spēlētāja ID ir 0..15, kamēr sesija ir aktīva. Bezsaistē tas ir -1.

```
id = whoami()
```

### nhost

`nhost()`

1, ja šī mašīna ir resursdators, pretējā gadījumā 0.

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

### nplayers

`nplayers()`

Cik tiešraides ID pastāv diapazonā no 0 līdz 15?

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

### nlisten

`nlisten()`

Sāk UDP mitināšanu un atgriež LAN adresi kā ip:port.

```
addr = nlisten()
```

### njoin

`njoin(addr)`

Pievienojas ip vai ip:port.

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

### nleave

`nleave()`

Iziet no sesijas. Līdzinieki redz _mp_leave ar iemeslu 0.

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

### nsend

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

Nosūta zudumradošu ziņojumu. Nosaukums ir notikums. Seko līdz četrām papildu vērtībām. p ir no 0 līdz 15 čuksti šim vienīgajam lietotājam; izlaidiet p, lai sūtītu visiem.

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

### ncall

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

Tā pati forma kā nsend, bet ziņojums tiek mēģināts atkārtoti, līdz tas tiek apstiprināts. Trīs kļūmes pēc kārtas pārtrauc sesiju.

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

### nlan

`nlan()`

Pašreizējā lokālā tīkla ip:port adrese pēc nlisten.

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

### nstun

`nstun()`

Jūsu publiskais ip:ports no STUN. Viesis citā tīklā var mēģināt tam pievienoties.

```
pub = nstun()
```

### setturn

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

TURN — rezerves risinājums, ja tiešs ceļš neizdodas. setturn() ar tukšiem argumentiem to izslēdz. Adrese ir resursdators:ports.

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

### nlist

`nlist(game)`

Atvērto sesiju tabula šim spēles nosaukumam. Katram ierakstam ir adrese, spēlētāji un vārds.

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

### nannounce

`nannounce(game)`

Sirdsdarbība, lai šis saimniekdators parādītos spēles nlistā. Izsauciet to taimerī, kamēr saimniekdators to vada.

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

### nkb

`nkb([on])`

Adreses lauks. nkb(1) to atver, nkb(0) to paslēpj, nkb() norāda, vai tas ir atvērts. Augšup un lejup vērstas bultiņas maina rakstzīmi, 0–9, punktu vai kolu. A to ieraksta. X dzēš. Tastatūra tajā neraksta.

```
nkb(1)
```

### nkbstr

`nkbstr([s])`

Mainās teksts adreses laukā, pēc tam rakstzīme augšpusē un apakšā. nkbstr("") notīra tekstu.

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

### _mp

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

Jūs rakstāt šo funkciju. Tā tiek palaista, kad pienāk ziņojums. "from" ir sūtītāja ID. "name" ir notikums. Seko līdz četrām vērtībām.

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

### _mp_join

`_mp_join(p)`

Jūs rakstāt šo funkciju. Līdzinieks saņēma ID p. Tā arī tiek aktivizēta jūsu vietā; whoami() ir šis ID.

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

### _mp_leave

`_mp_leave(p, why)`

Jūs rakstāt šo funkciju. Vienādranga partneris p iziet. Kāpēc 0 ir normāla aiziešana. Kāpēc 1 ir zaudēts savienojums.

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

### _mp_update

`_mp_update()`

Jūs rakstāt šo funkciju. Tā darbojas ik pēc 15 _update kadriem, aptuveni 4 Hz. Laba vieta, kur nosūtīt pozīcijas.

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

### _mp_fail

`_mp_fail()`

Jūs rakstāt šo funkciju. Sesija nedarbojas: resursdators ir pazudis vai ncall trīs reizes neizdevās.

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

## Korutīnas

### cocreate

`cocreate(f)`

Izveido korutīnu no funkcijas f. Tā nedarbojas, kamēr netiek izmantots coresume.

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

### coresume

`coresume(c, ...)`

Palaiž korutīnu c, līdz tā iegūst vai atgriež vērtību. Veiksmīgas darbības gadījumā: "true" un iegūtās vērtības. Kļūdas gadījumā: "false" un kļūda.

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

### costatus

`costatus(c)`

"apturēts", "darbojas" vai "miris".

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

### yield

`yield(...)`

Atdod kontroli atpakaļ tam, kurš izsauca coresume. Vērtības kļūst par šī coresume rezultātiem.

```
yield(1)
```

## Mērs

### info

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

Kad sesija ir beigusies, uzzīmē tiešraides HUD: fps (vidēji 1 sekunde), upd, drw, px, max, ram, peak, ramc un net. info() izmanto drukas kursoru. info(col) iestata tikai krāsu. info(x, y) vai info(x, y, col) to novieto. Skaitļi izmanto B, k vai m, lai tie ietilptu mazā ekrānā. Pats izsaukums netiek skaitīts pikseļos.

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

### showstats

`showstats([on])`

Tāds pats HUD kā info, automātiski uzzīmēts pie 0, 0 pēc _draw, ignorējot kameru un klipu. showstats(0) to paslēpj. showstats(1) vai showstats() to parāda.

```
showstats(1)
```

### perf

`perf(n)`

Viens HUD skaitlis, lai jūs varētu to uzzīmēt pats. 0 kadri sekundē, 1 atjauninājums, 2 zīmēšanas reizes, 3 pikseļi, 4 maksimums (smagākais kadrs šajā sekundē), 5 RAM, 6 RAM (RAM plus iepakotā spēles kasetne), 7 maksimālā RAM atmiņa kopš spēles kasetnes palaišanas.

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