PIX37 kasutusjuhend

Esmalt toimetaja ja siis kood. Raamatu võrguühenduseta koopia on saadaval aadressil pix37.party/docs.

Toimetaja

PIX37 DEV on redaktor. Siin joonistad, kirjutad ja kuulad mängu ning seejärel ekspordid mängukasseti või eraldiseisva versiooni. See saab ka mängukassetti mängida ja avada PIX37.PARTY, et alla laadida mängukassette ja kasutada kataloogi. PIX37 DEV EDUCATIONAL on sama redaktor ja see on tasuta. Sellel pole versioone, mängukassettide eksporti ega juurdepääsu PIX37.PARTY-le. PIX37 Console on mängija. See mängib `.p37` mängukassette ja avad PIX37.PARTY, et alla laadida mängukassette ja sirvida kataloogi.

Projektid ja mängukassetid

`.pix37` fail on projekt: lähtekoodi vahekaardid, спрайтlehed, kaardid, heli ja andmetabelid. `.p37` fail on mängukassett, mida mängija käitab: tihedalt pakitud baitkood.

Salvestused ja saavutused asuvad autori kaustas rakenduse andmete kõrval. Windowsis näeb see salvestuse puhul välja nagu `C:\\Users\\Ada\\AppData\\Roaming\\PIX37\\Mira\\Pong.p37s` ja saavutuste puhul `C:\\Users\\Ada\\AppData\\Roaming\\PIX37\\Mira\\Pong.ach`. Linuxis on samad failid `~/.local/share/PIX37/Mira/Pong.p37s` ja `~/.local/share/PIX37/Mira/Pong.ach`. Ada on arvuti konto. Mira on autori nimi ja see nimi on kaust. Pong on salvestuspesa, mis vaikimisi on mängu pealkiri. `save` kirjutab `.p37s`. Esimese avatud saavutuse puhul kirjutatakse `.ach`. Kui failid on juba olemas, loeb jooksja need mängu kasseti käivitamisel.

Arendajamenüü

Escape avab arendusmenüü. Mängu ajal Escape'i klõps viib tagasi redaktorisse. Kinnitage sisestusklahviga või nupuga A (vaikimisi V). Tagasi on nupp B (vaikimisi C).

Kui projekti pole laaditud, on menüü järgmine: CREATE PROJECT, LOAD PROJECT, BUILD, LOAD Game Cartridge, CONNECT TO PIX37.PARTY, SETTINGS.

Kui projekt on laaditud, on see järgmine: SAVE PROJECT, LOAD PROJECT, CREATE PROJECT, ASSETS, BUILD, LOAD Game Cartridge, CONNECT TO PIX37.PARTY, SETTINGS.

Ctrl+S salvestab avatud projekti. Ctrl+R käivitab selle.

Loo projekt

CREATE PROJECT küsib enne projekti loomise alustamist mängukasseti.

- **VIDEO** — 1-BITINE (2 värvi), 2-BITINE (4), 4-BITINE (16), 8-BITINE (256).
- **AUDIO** — süntesaatori sämpli sügavus: 16, 8, 4, 2 või 1 bitti.
- **EKRAN** – eelseadistused on järgmised: 64 × 64, 128 × 128, 160 × 90, 256 × 256 ja 320 × 180. KOHANDATUD suuruse abil saate valida mis tahes suuruse vahemikus 1 × 1 kuni 512 × 512.
- **PALETID** — värvipangad. Esimene pank alustab allpool oleva 16-värvilise komplektiga. LOAD PALETTE rakendab sisseehitatud komplekti: PICO-8 16, PICO-8 32, Game Boy, 1-bitine, CGA, Sweetie-16 ja pikem nimekiri Lospecist (Resurrect 64, Apollo, Endesga 32 ja 64, CC-29, SLSO8, Pear36, Oil 6, Duel, AAP-64, Lost Century ja teised). Samuti saate laadida paletifaili või oma PNG-faili ja võtta paleti sellest pildist.



`cls`, `pset`, `spr` ja teiste poolt kasutatavad värviindeksid. Paletiredaktoris saab mis tahes indeksit ümber värvida.

Viis paneeli

Redaktor on GAME, PAINT, CODE, SOUND ja DATA, selles järjekorras. Kaart asub PAINTi sees. Heliefektid ja laulud asuvad SOUNDi sees. Klõpsa paneelil, et seal töötada. Kui CODE on fookuses, ignoreerivad SOUND ja DATA täheklahve, seega tippimisel ei mängita samal ajal klaverit.



Täispaigutus, kirjapilt `1:1:4:1:1`. Iga number näitab, kui suure osa reast paneel võtab: MÄNG, VÄRVIMINE, KOOD, HELI, DATA. Väike ekraan asub 128×128 lahtri keskel. Ekraan, mis on suurem kui 128 pikslit, laiendab MÄNG

veergu.

SEADED → GLOBAALNE → PAIGUTUS vahetab paigutust:

Kood	Mida sa näed
`1:1:4:1:1`	Kõik. MÄNG ja HELI koos, VÄRVIMINE ja DATA koos, KOOD paremal.
`1:1:4:1:1 2X`	Sama paigutus. KOOD joonistatakse kahekordse tihedusega, seega mahub sinna kaks korda rohkem teksti.
`1:0:1:0:1`	MÄNG, ANDMED, KOOD.
`1:0:1:0:1 2X`	Sama paigutus, tihedama KOODIGA.
`0:1:1:1:1`	Mängu eelvaadet pole. VÄRV, KOOD, HELI, ANDMED.
`0:1:1:1:1 2X`	Sama paigutus, tihedama KOODIGA.
`0:1:4:1:2`	Stuudio. Lai CODE, kõrgem DATA riba, PAINT ja SOUND.
`0:1:4:1:2 2X`	Sama paigutus, tihedama KOODIGA.
`0:0:4:0:4`	Ainult KOOD ja ANDMED, mõlemad suured.
`0:0:4:0:4 2X`	Sama paigutus, tihedama KOODIGA.
`0:1:4:1:0`	VÄRV, HELI ja lai KOOD.
`0:1:4:1:0 2X`	Sama paigutus, tihedama KOODIGA.
`0:4:0:0:0`	Ainult VÄRVIMINE, 256×256, üks lehe piksel ühe lõuendi piksli kohta.

`2X` rida säilitab samad paneelid. CODE paneel joonistatakse kahekordse eraldusvõimega ja skaleeritakse tagasi samasse lahtrisse, seega on tekst, tabulaatorid ja äärised poole kõrgemad ning ekraanile mahub kaks korda rohkem lähteteksti.

MÄNG

Mängukasseti eelvaade. Eelvaate käivitamiseks klõpsake MÄNG paneelil või vajutage Ctrl+R. Paneeli konfigureerimiseks valige SEADED → MÄNGUPANEELI.

- **Eelvaade** – värviline, pidev või automaatne uuestilaadimine redigeerimise ajal.
- **Mängi uues aknas** — mäng avaneb redaktori kõrval.
- **Ekraan** – eelvaade kui RGB888 või RGB565.
- **MÄNG 64** — 64×64 ekraan on kas lahtri keskele 1:1 tsentreeritud või skaleeritud 2×, et see täita.
- **Varjundid** – CRT ja filtrid mängus, eraldi redaktori filtritest.
- **Värv** — paneeli taust.

VÄRV

Spriteeri lehed ja sama paneeli kaardipoolel paanidega kaardid. Lehed võivad olla ekraanist suuremad. Spriteeri eelvaate suurused on 8, 16, 32, 64 ja 128 (128 ainult siis, kui lehe suurus on vähemalt 128 pikslit). Suumimiseks kerige hiireratast spriteeri eelvaate kohal. Pintsli suurused on 1, 2, 3 ja 4.

Võti	Tööriist
B	Pliiats
M	Ristküliku valimine
K	Lasso
S	Tempel
G	Ämber
Mina	Pipett
V	Liigu
L / Shift+L	Joon / kõver
U / Shift+U	Õõnes ristkülik / õõnes ellips
F / Shift+F	Täidetud ristkülik / täidetud ellips
D	Kõhklemine
[ ]	Väiksem/suurem hari
Ctrl+Z / Ctrl+Y	Võta tagasi / tee uuesti. Ctrl+Shift+Z teeb samuti uuesti.
Ctrl+C / Ctrl+X / Ctrl+V	Kopeeri, lõika, kleebi
Ctrl+A	Valige lõuend

Kaart kasutab sama ideed, vähemate tööriistadega: B pliiats, M ristküliku valik, S tempel, V liigutamine ja Ctrl+Z, Ctrl+Y, Ctrl+C, Ctrl+V. Kaardi tekstuuri suumimiseks hoidke all Alt. Pintsli suurus kaardil jääb värvitööriistade suuruseks.

Igal спрайdil on kaheksa lipubitti. Paint määrab need спрайdile; `fget` ja `fset` loevad neid mängus.

SEADED → PAINT EDITOR on koht, kus seda paneeli konfigureeritakse:

- **Palett** — milline pank joonistatakse, pluss laadimine ja salvestamine.
- **Eelvaade** — RGB888 või RGB565.
- **Pintslikursor** – osuti maalimise ajal.
- **Pikslitõmme** — kuidas pintsel pikslitele klõpsub.
- **Suur pintsel** – suur pintsel klõpsab vasakust ülanurgast või kogu oma jalajälje ulatusest.
- **Sibulakoor** — teiste raamide kummitus ja millises suunas see kummitus vaatab.
- **Kaardi suum** — tõelised pikslid ehk vaba suum.
- **Värvid** — värvi kasutajaliidese värvid.

KOOD

Vahekaardid on projekti sees olevad eraldi failid. Vahekaardi nimel klõpsates saab selle ümber nimetada; sisestusklahvi vajutamine kinnitab. Mitu vahekaarti, mis igaüks määratlevad `_init`, `_update` või `_draw`, on kokku liidetud vahekaartide järjekorras: esimene vahekaart kuvatakse esimesena, viimane vahekaart viimasena. Kood, mis ei asu konksu sees, saab osaks `_init`-st. Mängukassetti, millel pole ühtegi konksu, käsitletakse ühe `_draw`-na.

Võti	Tegevus
Nooled	Liiguta kursorit. Shift laiendab valikut.
Ctrl+vasak/parem nool	Sõna järgi.
Home / End, Ctrl+Home / End	Rea algus või lõpp või tabeldusmärgi algus.
Vahekaart	Lisa kaks tühikut. Aktsepteeri nimi loendis.
Sisesta	Uus rida, taandatud nagu eelminegi. Aktsepteerib rivi täiendamist.
Ctrl+C / X / V	Kopeeri, lõika, kleebi.
Ctrl+Z / Ctrl+Y	Võta tagasi / tee uuesti.
Ctrl+A	Valige vahekaart.
Ctrl+D	Dubleeri joon või valik allapoole.
Ctrl+/	Kommenteeri või tühista kommentaar koodiga `--`.
Ctrl+F	Leia. F3 hüppab järgmise klõpsu juurde. Escape sulgeb riba.
Ctrl+G	Mine rea numbri juurde.

Põgenemine	Sulgeb esmalt otsingu, mineku või vihjete loendi. Teine Escape-klahv lahkub redaktorist ja avab arendusmenüü.

Funktsiooni nime tippimise ajal kuvatakse vihjeid. Enter või Tab sisestab nime ja koodi `(`, kui sätetes on sisse lülitatud COMPLETE (). SEADED → KOODIREDAKTOR:

- **Otsi** – Ctrl+F aktiivsel vahekaardil või igal vahekaardil.
- **Teksti kõrgus** – kärpige glüüfe 5 pikslini või kuvage kogu glüüf.
- **Stringi suurus** — stringi literaalide fondi suurus.
- **Kommentaari font** — 3×5 või praeguse kasutajaliidese keele Fusioni font.
- **Täidetud (** — kas lõpetamine tüübib sulgu.
- **Koodivihjed** — sees või väljas.
- **Vihjete keel** — menüükeel ehk inglise keel.
- **Sisestatav vahe** – tühjadel ridadel täiendav nihe.
- **Vihmaveerennid** — reanumbrid.
- **Värv** – taust, kommentaarid, tabulaatorid, numbrid, operaatorid, märksõnad, stringid, mootori funktsioonid, teie funktsioonid, muutujad, sulud, kaar, valik, vearida, hoiatus.

HELI

Paneelil on kaks vahelehte. SFX on üks muster: lühike nootide jada. SOUND on laul: see paigutab need mustrid kaheksale kanalile. TIC näitab, mitu tikki üks samm ootab. Kell on 120 tikki sekundis ja üks samm on kuueteistkümnendiknoot, seega 15 tikki on 120 lööki minutis. Kõrgem TIC on aeglasem muster. Kaks mustrit püsivad samas taktis, kui üks TIC jagab teist: 15 ja 30 jäävad, 15 ja 8 mitte.

Klaver mängib, kui hiirega noot valitakse. Valged klahvid on ZXCVBNM (CDEFGAB). Mustad klahvid on SDGHJ (C# D# F# G# A#). Liigutage kursorit mis tahes väärtuse kohale ja tippige number või liigutage kursorit selle kohale ja kerige ratast. Koodis on helikõrgused vahemikus 0 (C-0) kuni 63 (D#5).

Ühel mustril on LOOP ja LEN samad kaks numbrit. OUT üle 0 kordab IN-st läbi OUT-i ja pealkirjas on LOOP. OUT 0 juures ja IN üle 0 on LEN: nii palju noote mängib algusest peale, seejärel muster lõpeb ja pealkirjas on LEN. Mõlemad 0 juures mängivad iga sammu.

See, kui palju süntesaatorit saab redigeerida, sõltub mängukasseti jaoks valitud bitisügavusest. SEADED → SFX-REDAKTOR määrab selle sügavuse, kas noot liigub automaatselt edasi (väljas, noodiahel või väljaahel), kas tühikuklahviga taaskäivitatakse taasesitus või lülitatakse käivitamist ja peatamist ning redaktori värvid.

16-bitine

Igal sammul on noot, helitugevus, laine ja efekt. Helitugevus on 0 kuni 7. 0 on hääletu.

Lained:

- **0 kolmnurk**
- **1 kallutatud sae**
- **2 saega**
- **3 ruutu**
- **4 pulss** — kitsas pulss
- **5 organit**
- **6 müra**
- **7 faasijat**
- **8 PWM** — ruut, mille laius liigub
- **9 pulssi 25%**
- **10 siinust**
- **11 metallilist müra**
- **12 FM kell**
- **13 kitkumist**
- **14 supersaag**
- **15 PWM (C64)**

Efektid ühel noodil:

- **0 mitte ühtegi**
- **1 slaid** – helikõrgus ja helitugevus libisevad sellel sammul järgmisele noodile
- **2 vibrato** — helikõrgus võngub
- **3 drop** — heli langeb vaikusesse, tavapärane löök
- **4 sujuv sissetulek**
- **5 hääbub**
- **6 arpedžo** – kõnnib sellel ja järgmistel kolmel noodil iga 8 tiki järel. TIC 8 või alla selle korral iga 4 tiki järel.
- **7 kiiret arpedžot** — samad neli nooti iga 4 tiksu järel või iga 2 järel, kui TIC on 8 või alla selle.

Mustri päis märkmete kohal:

- **TIC** – tikid sammu kohta, 1 kuni 255. 15 on 120 lööki minutis. Tippige number või samm-sammult.
- **LOOP IN / OUT** – kui OUT on suurem kui 0, siis muster silmuses IN-st OUT-i. Jäta OUT väärtuseks 0 ja pealkirjaks saab LEN: IN tähistab mängitavate nootide arvu ja 0 tähendab kogu mustrit.
- **MÜRA** — kogu muster muutub müraks.
- **CRUSH** — iga laine elav variant.
- **DTUNE** — 0 välja lülitatud, 1 kerge refrään, 2 tugev refrään.
- **RVERB** — kaja: 0 välja lülitatud, 1 lühike, 2 pikk.
- **DAMPN** — madalpääsfilter: 0 välja lülitatud, 1 nõrk, 2 tugev.
- **TASE** — kogu mustri helitugevus, 0 kuni 7.
- **KIIK** — lohistamine. Paaritud sammud ootavad veidi kauem, ühtlased sammud veidi vähem.
- **SAMMUD** — mitu sammu muster kasutab: 32, 24, 16 või 12.

8-bitine

Sammul on noot ja laine. Neli lainet on ruut-, kolmnurk-, saag- ja müralained. Päises on TIC, LOOP IN/OUT, FX ja LEVEL. TIC on üks järgmistest: 1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 16, 20, 24, 30, 40, 60. FX on üks efekt kogu mustri jaoks: 0 puudub, 1 vibrato, 2 arpedžo, 3 hääbub. LEVEL on 0 kuni 3.

4-bitine

Samm on ainult noot. Päises on TIC, LOOP (sisse või välja lülitatud), WAVE (samad neli lainet) ja OCT (millises oktaavis noodid asuvad). TIC on üks järgmistest: 4, 6, 8, 10, 12, 15, 20, 30.

2-bitine

Samm on tühi, trumm, löök või purustus. Päisel on TIC ja BASE. TIC on 8, 15, 16 või 30. BASE määrab, kui kõrgel trumm ja löök asetsevad.

1-bitine

Samm on noot. TIC on 8, 15, 16 või 30, samad neli kiirust mis 2-bitisel.

Laulud

Vahekaart „HELI" tähistab avatud lugu. Kaheksa veergu tähistavad kanaleid. Lahter tähistab mustri ID-d. Tühjaks joonistatud 255 tähendab, et kanal on sellel real vaikne. X-veerg tähistab käsku: jätka, peata, tsükli algus või tsükli lõpp. „Stopp" lõpetab loo rea järel. „Tsükli lõpp" hüppab tagasi viimase tsükli algusesse.

Rida lõpeb siis, kui vasakpoolseim muster, mille LOOP OUT on 0, lõpeb. Parempoolsemad mustrid seda rida ei pikemaks tee: pikem muster lõigatakse läbi, lühem alustab uuesti oma esimesest sammust kuni rea lõpuni. Muster, mille LOOP OUT on üle 0, teeb silmuse LOOP IN ja LOOP OUT vahel ega lõpe rida. Kui iga muster real teeb silmuse, jääb rida sinna.

Otseteed

Võti	Tegevus
Klaveriklahvid	Kirjuta hiirega valitud sammu kohta märkus.
Üles / Alla	Muutke kursori all olevat väärtust.
Kosmos	Esita avatud heliefekti või esimese rea avatud lugu. Alusta/peata või taaskäivita heliefekti redaktori seadetest.
Ctrl+Z / Ctrl+Y	Võta tagasi / tee uuesti.

Ctrl+C / Ctrl+X / Ctrl+V	Kopeeri, lõika, kleebi valik.

ANDMED

Nimega tabelid. Iga tabel koosneb ridadest ja veergudest. Koodist `dget`, `dset`, `dstr`, `drows` ja `dcols` kasutavad neid tabeleid. Rea- ja veerunumbrid algavad koodis 0-st.

Veergude tüübid, päises tsükliliselt kuvatud:

- **bool** — üks bitt.
- **2** — kaks bitti.
- **4** — neli bitti.
- **8** — bait.
- **16** — 16 bitti ID-de ja suuremate arvude jaoks.
- **spr** — спрайdi ID.
- **str** — tekst, mis on joonistatud süsteemifondiga.
- **str8** ja **str10** — sama tekst, ainult kõrgemaks joonistatud.
- **f16** ja **f32** — pool-täpsusega ujukombo ja täis-ujukkombo.

Kui kursor on veeru päisel, muudab Ctrl+kerimine selle veeru tüüpi. Tavaline kerimine liigub üle veergude või ridade alla, olenevalt sellest, kumba SEADED → ANDMEREDAKTOR valib. Ctrl vahetab seda suunda, mistõttu Ctrl+kerimine päisel muudab tüüpi, kui ratas on veergudele seatud.

Nooleklahvid liiguvad. Enter redigeerib lahtrit. SEADED → ANDMEREDAKTOR:

- **Ratas** – veergude või ridade kerimine.
- **Värvid** — ruudustik, päis ja valik.

Varad

VARAD impordib ja ekspordib avatud tüki, mitte tervet projekti. Sprite'id kasutavad PAINT-is avatud спрайти avatud lehel. Kaart on avatud kaart. SFX on avatud muster. Muusika on avatud laul. Andmed on avatud tabel. Kood on erand: iga tabel kirjutatakse ja iga tabel loetakse tagasi.

Lahke	Fail	Mis liigub
Sprite'id	`.png`	Avatud sprite.
Kaart	`.p37map`	Avatud kaart.
Heliefektid	`.p37sfx`	Avatud muster.

Muusika	`.p37mus`	Avatud lugu. Mustrite numbrid jäävad samaks. Ekspordi heliefektid eraldi, kui neid helisid vajad.
Andmed	`.p37dat`	Avatud laud.
Kood	`.lua`	Iga vahekaart.

Seaded

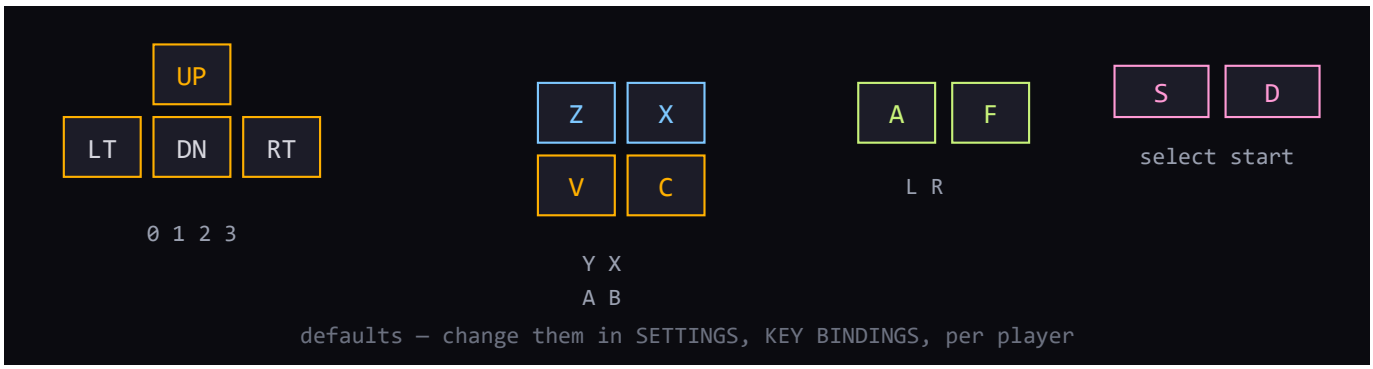
Juurte loend:

Leht	Mida see salvestab
PROFIILID	Salvesta ja laadi kõik sätted. Loo uus, nimeta ümber, kirjuta üle, kustuta. Kopeeri või kleebi profiil tekstina.
Menüü	Aken, klahvid, menüüvärvid, valikuline taust.
MÄNGUPANEELI	Eelvaade ja esitusaken.
KOODIREKTAATOR	Esiletõstmine ja vihjed.
VÄRVI TOIMETAJA	Pintsel, palett, kaardivaade.
SFX-redaktor	Süntesaatori ja noodi sisestamine.
ANDMETE TOIMETAJA	Rataste suund ja värvid.
KEEL	Menüü ja vihjete keel.
LITSENTSID	Litsentsitekstid, sh font Fusion Pixel ja SDL3.

GLOBAALNE:

- **PAIGUTUS** — paneelide paigutus, ülal.
- **HELIMAHT** — meistritase, 0 kuni 10.
- **KLAHVI SEGADUSED** — klahvid igale 8 mängijale. Klahvi löömiseks vajutage seda. Juhised, ABXYLR, vali.
- **Present (olevik)** – jäta vahele tühikäigul olevad kaadrid või joonista iga kaader.
- **Täisekraan** — ääristeta.
- **Värskenda** – akna kaadrisagedus. Vasak ja parem samm 60, 90, 120 ja maks. Sisesta tüüp, sisesta number kuni 1000-ni. 0 tähendab, et kaadrisagedust pole.
- **Vsync** — oota kaadrite vahel kuvamist.
- **Skaala** – täisarvuline akna kordaja.
- **Varjutajad** — CRT ja filtrid redaktoris, mitte mängus.

- **Värv** – menüütekst ja esiletõstmine.
- **Taust** – valikuline PNG-fail, kärbitud 128 piksli suuruseks menüü taga. Nupp „Tühjenda“ eemaldab selle.
- **Redaktori kursor** — pointeri kuju.
- **Hiir** – ujuv kursor või pikslitele klõpsitud.
- **Kasutajaliidese kiirus** – kui kiiresti hoitakse suunda loendis korduvalt.
- **Explorer** – faililoendid algavad arvuti juurkaustast või exe-faili kaustast.



Vaikimisi klaviatuur. A on menüüdes kinnitusnupp. B on tagasinupp. Mängus on need nupud 4 ja 5. Vali (S) on nupp 10, mitte menüü otsetee.

Nupp	Roll	Vaikimisi võti
0	Vasakule	Vasaknool
1	Parem	Parem nool
2	Üles	Ülesnool
3	Alla	Allanool
4	A, kinnitage	V
5	B, tagasi	C
6	X	X
7	Jah	Z
8	R	F
9	L	A
10	Vali	S
11	Alusta	D

Ehitamine

Valige platvorm: Windows, Linux, Mac, Android, iOS või veeb.

- **Nimi** — ehituse failinimi.
- **Autor** – Windowsis ja Linuxis. See on salvestuste kausta nimi.
- **Steami rakenduse ID** – Windowsis ja Linuxis. Tühi tähendab, et Steamworksi pole. Täidetud ID seob `achieve(0)`…`achieve(15)` Steami saavutustega ACH_0…ACH_15.
- **Id** – paketi identifikaator, kui platvorm seda küsib.
- **Peida kood** — lauaarvuti ja mobiili ülekattega ekraanidel. See aeglustab mängukasseti `peek`. See ei ole lukk.
- **Täisekraan** – eksporditud mängija käivitub täisekraanirežiimis.
- **Varjundid** – kuni kolm kihti mängijasse sisse ehitatud.
- **Laadi alla .p37** – ainult veebis. Kuvab pausimenüüs allalaadimise kirje.
- **Ikoon** – sprite'i lehelt pärit paan: leht, suurus, x, y. Või kärbitud PNG-kaanepilt.
- **Ekspordi** – valige kaust. Menüüs kuvatakse teade „EHITAMINE SALVESTATUD“ või „EHITAMINE NURJUNUD“.

Loo internetiühendusega ehitis. Avaleht võib serverilt anonüümselt küsida, kas selle ehitise räsi on teada. See kontroll on olemas kahel põhjusel. See kaitseb mind, PIX37 autorit, tööriista koopiate eest, mis ei ole minu omad, et neid ära anda. See kaitseb ka mängu loojat. Kellegi teise mängu võtmine ja enda omaks märkimine on telefoniturul, näiteks Play (Android) või iOS-is, muutumas tavaliseks. Mäng ikkagi algab. Kui masin on võrguühenduseta, ütleb intro, et mäng pole ühendatud. See rida on ka vihje mängijale, et koopia ei pruugi pärineda legaalsest allikast. See on väike kontroll, nii et kui teie mäng varastas ja müüs keegi teine ilma teie nõusolekuta, saate selle juhtumi esitada poes, näiteks Play (Android), Steam või itch.io. Server salvestab räsi esmase registreerimise aja räsi kõrvale ja ei muuda seda hiljem. Varasem aeg on algne ehitis. Pärast seda registreeritud räsi on hilisem koopia.

Ma tahan, et see kontroll kaitseks mänge loovaid inimesi. Mängumootor teeb juba suure osa tööst nende eest ära ja see on üks neist asjadest. Ma saan aru, et see võib olla vastuoluline. Kui kogukond otsustab, et see mikro-DRM on parem ära jätta, siis ma teen seda.

Eelmisest redaktorist pärit vana mängukassett võib pärast keele muutmist keelduda käivitumast. Ava selles redaktoris projekt `.pix37` ja eksporti uus `.p37`.

Loo ühendus PIX37.PARTY-ga

ÜHENDUSE LOOMINE PIX37.PARTYGA toimub rakenduse sees. Brauseris sisselogimislehte ei ole. Sisse logitakse Steami seansi pileti või itch.io kaudu, kasutades tahvelarvutit ja hiirt. Teenus kuvab unikaalset 2–16 tähemärgist koosnevat kuvatavat nime. Litsentsivõtit ja Steami ID-d teistele mängijatele ei kuvata.

Pärast teenusetingimusi on keskuseks Lemmikud, Kataloog, Edetabel, Mängijad ja teie profiil.

Arendajalitsents on toimetaja. Sina lood mängu, laadid selle üles ja jätad tähed maha. Mängurilitsents on mängija. Südamed tulevad inimestelt, kes jooksuril mängivad.

- **Kataloog** – sirvi mänge. Filtrid hõlmavad žanri (kuni kolm), ekraani suurust, mängukasseti suurust ja seda, kas üleslaaditav fail sisaldab `.pix37` projekti.
- **Lemmikud** – mängud, mille olete märkinud.
- **Üleslaadimine** – vajab arendaja litsentsi. Saadad `.p37` ja võid lisada `.pix37` projekti. `.p37` on pakitud baitkoodiga fail. Seda on raske lugeda ja seda saab tahtlikult hägustada. `.pix37` on projektifail, millega sa tegelikult töötad: lähtekood, pildid ja heli kujul, mida teine inimene saab avada ja millest õppida. Selle saatmine võimaldab teistel loojatel mängu uurida pakitud mängukasseti asemel. Sellega kaasnevad pealkiri, autor, žanrid, kaanepilt, ikoon ja saavutuste nimed. Saadetav RAM-i number on mängukasseti saavutatud maksimaalne mälumaht.
- **Kommentaarid ja saavutused** – mängulehel. Saavutuse avamine sisselogituna salvestatakse nii sellele kontole kui ka masinasse.
- **Edetabel** – arendajad, sorteeritud tähtede või südamete järgi. Igas reas kuvatakse mõlemat arvu. Ava nimi, et näha selle autori mänge südamete järgi järjestatud.
- **Mängijad** — avatud saavutuste järgi järjestatud kontod.

Kood

Keel on väike Lua-stiilis skript. See pool alustab tühjast projektist, õpetab keelt ja seejärel näitab iga funktsiooni lühikest näidet.

Esimene programm

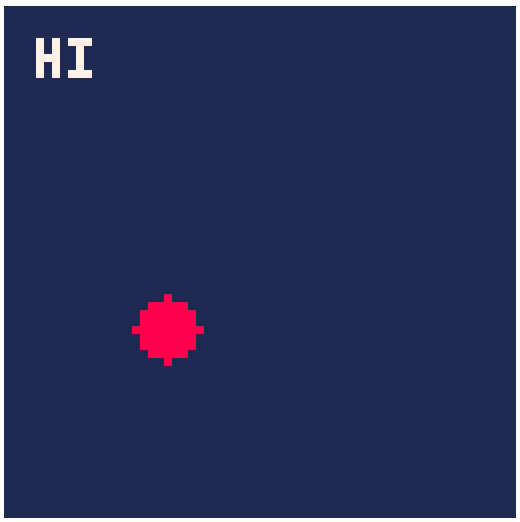
Loo projekt mõõtmetega 64×64, 4-bitise värviga. Ava CODE. Asenda tabeldusmärk järgmisega:

```
_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 käivitab selle. Nooleklahvid liigutavad ringi. Escape peatab mängu ja naaseb redaktorisse.



Samad kolm käsku: `cls(1)` värvib tumesinise ekraani, `print` joonistab HI ja `circfill` joonistab roosa ringi.

Mis jookseb ja millal

- `_init` käivitub ühe korra.
- `_update` töötab sagedusel 60 Hz. Pane siia liikumine ja nupud.
- `_draw` joonistab. Ilma kaadrisageduse argumendita joonistab see nii kiiresti, kui arvuti ja monitor suudavad. `_draw(30)` piirab joonistamise 30 kaadrisagedusega ja jätab loogika 60 peale.
- `_set_fps(n)` muudab seda ülempiiri hiljem. 0 tühjendab selle.

Nimed ei ole suur- ega väiketähtedega; redaktor salvestab need väiketähtedega. `function` ja `fun` on sama märksõna.

Keel

Programmeerimise põhitõed

Programm on ridade loend, mida mängukassett käivitab. Iga pealkiri näitab esmalt rea kuju ja seejärel väikest näidet. Rida, mis algab `--`-ga, on kommentaar: märkus sulle. Mäng jätab selle vahele.

Muutujad

Muutuja on nimi, mis sisaldab väärtust. Sina valid selle nime. Hilisemad read saavad seda lugeda ja salvestada sama nimega uue väärtuse.

Kuju on `name = value`. `local name = value` hoiab selle nime praeguses plokis, seega ülejäänud mängukassett seda ei näe. Ilma `local`-ta on nimi globaalne ja iga kaart näeb seda.

```
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` tähendab, et väärtust pole. `true` ja `false` on jaatavad ja eitavad. Nimi, mida te pole kunagi määranud, on `nil`.

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

Lauad

Tabel sisaldab ühe nime all palju väärtusi. Loend kasutab numbreid. Esimene element on 1, mitte 0. Kirje kasutab nimesid. Tabel võib sisaldada teist tabelit.

Loendi kuju: `{ value, value }`. Kirje kuju: `{ name = value, name = value }`. Kirjete loend on tabel, mille elemendid on tabelid.

```
-- 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
```

Seisund ja keha

Tingimus on küsimus, millele saab vastata jah või ei, näiteks `hp > 0` või `btnd(4)`. Põhitekst on ridade rühm, mis jookseb ploki sisenemisel. `if` ja iga tsükkel kasutavad sama kuju: tingimus, seejärel põhitekst ja seejärel `end`. `repeat` lõpeb `until`-ga `end` asemel.

Kui

`if` käivitab oma põhikoodi ainult siis, kui tingimus on tõene. `elseif` proovib teist tingimust. `else` käivitatakse siis, kui kõik tingimused on väärad.

Kuju: `if condition then`, seejärel keha ja seejärel `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
```

Silmused

Tsükkel kordab põhifunktsiooni. `while` kontrollib esmalt tingimust ja jätkab seni, kuni see jääb tõeseks. `repeat` käivitab põhifunktsiooni ja seejärel kontrollib, nii et põhifunktsioon töötab alati vähemalt ühe korra. `for` loendab. Viimane number on kaasatud. Ilma kolmanda numbrita on samm 1, seega kui algus on juba lõpust möödas, siis põhifunktsiooni ei käivitata. Allapoole loendamiseks on vaja negatiivset sammu.

```
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
```

Laua ääres kõndimine

`all` külastab loendi iga elementi alates esimesest. `pairs` külastab iga nime ja selle nime alla salvestatud väärtust.

```
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
```

Funktsioonid

Funktsioon on nimega retsept. Kuju on `function name(arguments)`, seejärel keha ja siis `end`. Seda kutsutakse nime ja sulgude kirjutamise teel. `return` saadab tagasi väärtuse ja see võib saata rohkem kui ühe. Argumendid on selle funktsiooni lokaalid.

```
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` on funktsioon, mis näeb enda ümber olevaid lokaale. See on sulgemislause. Tabel mahutab ühe sellise ja seda kutsutakse punktiga.

```
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
```

Jah, ei ja valimine

`not` pöörab tingimuse ümber. `and` on tõene ainult siis, kui mõlemad pooled on tõesed. `or` on tõene, kui kumbki pool on tõene. `a and b or c` valib `b`, kui `a` on tõene, vastasel juhul `c`. `cond ? a : b` on sama idee ühel real: `a`, kui tingimus on tõene, `b`, kui see ei ole.

```
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)
```

Matemaatika ja tekst

`+ - * /` on liitmine, lahutamine, korrutamine ja jagamine. `//` jagab ja jätab täisarvu alles. `%` on jääk. `^` on aste. `..` liidab kaks teksti. `#t` näitab loendi üksuste arvu. Võrdlused on `==` (sama), `~=` või `!=` (erinev) ja `< > <= >=`. `x += 1` liidab x-ile 1. Sama kujund töötab ka `-= *= /= ^= %=` puhul.

```
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
```

Kus kohalik elab

`local` kestab kuni `end`-ni, mis selle ploki sulgeb. `if` / `end` on omaette plokk. Võite mainida lokaali rea kohal, mis seda määrab, kuid kuni selle rea käivitamiseni on väärtus `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
```

Hüppamine

`::label::` tähistab rida. `goto label` hüppab sinna, aga ainult sama funktsiooni sees.

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

Mälu, lühidalt

`peek` ja `poke` vaatavad reaalajas piksleid ühe 16-bitise aadressi kaudu. Nad ei hoia mängukassetist teist koopiat. Aadressid `0x5F00` kuni `0x5FFF` on registrid: kaamera, klipp (kirjutuskaitstud), joonistusvärv, täitemuster, kolm paleti ümberkirjutust ja leheküljenumbrid. Selle augu mõlemal küljel olevad baidid, mida on 65280, moodustavad piksliaken.

Kui ekraan mahub nendesse 65280 baiti (64×64 ja 128×128 mahuvad ära ning 256×256 mitte), algab ekraan rida-realt aadressilt 0. Järele jääb `0x5F41` poolt valitud sprite'i leht (0 on leht 0). Kui see leht on suurem kui allesjäänud ruum, valib `0x5F42` lehe ja kaarti aknasse ei tule. Kui leht mahub ära, on saba `0x5F43` poolt valitud kaardi lahtrid, igaüks üks või kaks baiti, ja `0x5F44` lehekülgistab kaardi, kui see sabasse ei mahu.

Kui ekraan ei mahu, on kogu aken üks 65280 piksliga leht. `poke(0x5F40, n)` pöörab ekraani lehekülge. `poke(0x5F41, n)`, mille n on muu kui 0, näitab hoopis lehte n ja lehel 0 pole selles režiimis vaadet. `poke(0x5F43, n)`, mille n on muu kui 0, näitab kaarti n ja see võidab lehe. Kaart 0 on selles režiimis ainult `mget` ja `mset`.

`peek`, mis ei satu pikslile ega registrile, tagastab 0. `poke` ei tee midagi. Sprite'i lipud, SFX ja DATA tabelid selles aknas puuduvad. Kasutage `fget`, `sfxget` ja `dget`.

Praktiline kasutamine

Kõik funktsioonid on loetletud selle lehe allosas koos PIX37 funktsioonide praktiliste kasutusvõimalustega. Iga näite all olevad lingid avavad funktsioonide kohta lisateavet.

Sujuv liikumine

128×128 mängukassett ülaltvaates. Sprait 1 on juba joonistatud. Nooled kiirenevad igas suunas ja hõõrdumine aeglustab sind. Sprait mähitakse: pool, mis lahkub alt, joonistatakse üles ja sama kehtib ka teiste servade kohta.

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

Kontrollitakse, kas järgmine käik on blokeeritud

64×64 mängukassett. Põrand ja parempoolne sein on juba kaardil, värvitud спрайдтиga 3 ja sellele спрайдile on seatud lipp 0. Nooled liigutavad спрайдti 1. Üleval olev sõna ütleb „blokeeritud", kui järgmine samm kattuks täisruuduga, ja „vaba", kui see ei kattuks.

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

Pühkimine klapiga

Vajuta A. Pühkimisriba joonistatakse flip() abil, üks must rida korraga, mitte _draw() sees. _draw jääb selle töötamise ajal eest ära, seega jääb iga rida ekraanile.

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

Funktsioonide tabel (klass)

Sprait 1 on juba joonistatud. make_monster töötab nagu klass. See tagastab tabeli ning update ja draw on meetodid, mis säilitavad selle koletise enda x ja y väärtused. Ülaosas olev number näitab, kui kaugele see on kõndinud.

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

Neli viisi nupu lugemiseks

A on nupp 4. Hoia seda all ja laengu kiirus tõuseb. Vabasta see ja laengu kiirus langeb tagasi 0-ni. Hoidmise ajal korduvad üles ja alla liikumised, nii et menüürida liigub pidevalt. btnd on vajutuse esimene kaader, mida see mängukassett kasutab ainult all hoidmise algusena.

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

Tulista hiirega

Sprait 1 on juba keskele joonistatud. Valge täpp on kursor. Vasak klõps suunab piksli selle poole. Rullik muudab kiirust ja see on trükitud nurka.

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



Tippige nimi ja kinnitage see

Välja avamiseks vajutage A. Üles ja alla nooleklahvid muudavad vilkuvat tähte. A kirjutab selle, X kustutab, Y on tühik ja B kinnitab. Seejärel kuvatakse real teie nimi.

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



Kaamera mängijal

128×128 mängukassett. Sprite 1 algab plaatidest koosneva põranda keskelt. Nooled liiguvad. Kaamera hoiab sprite'i tsentreerituna. Nurgas olevad x ja y ei keri, sest argumentideta camera() paneb vaate tagasi HUD-i jaoks.

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



Mõned read visuaalsest romaanist

Jooned on DATA-vormingus, inglise ja poola keeles. A jätkub ehk valikud jäävad ekraanile, kui valik on kuvatud. B valib valiku pealt lahkumise ja muul juhul keele vahetamise. X vahetab keelt valiku kuvamise ajal. Dialoog kasutab 8px fonti, et poola tähed joonistaksid.

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



Värvige piilumise ja torkimisega

64×64 mängukassett, seega aadress 0 on ülemine vasakpoolne piksel ja järgmine rida algab 64-st. Mängukassett kirjutab piksleid, kopeerib rea, tühjendab rea ja prindib loetud väärtused.

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



Nimekiri, kuhu saate lisada

A liidab arvu kuni 8-ni. B eemaldab viimase. X eemaldab esimese. Loend prinditakse ekraani alla.

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



Nimi lahti rebitud

Mängukassett jagab arvud ada,12,blue osadeks ning prindib välja nime, selle esitähe 12 ja sõna blue.

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



Mängi noot ja muuda seda

Muster 0 on ühe noodi pikkune ja iga kordus on sama valjus. Muster A mängib seda uuesti, kui see noot on lõppenud. Muster B tõstab helikõrgust ja mängib selle noodi.

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



Mängukassetti kirjutatud laul

Real 0 kordub kaks mustrit, üks kummalgi kahel esimesel kanalil. Rida jääb 0-ks. ch0 ja ch1 tähistavad nooti, millel iga kanal asub. Mustri, LEN ja LOOP muutmine mängukassetis annab võimaluse näha, mida need juhtnupud muusikapaneelil teevad.

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



Kolm koletist DATA-s

ANDMETE paneelil on juba nime veerg ja elujõu veerg koos lima, nahkhiire ja silmaga. A teeb limale haiget. Parempoolsed numbrid näitavad selle elujõudu.

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



Pea meeles skoori

A lisab punkti ja salvestab selle. Väljub ja avab mängukasseti uuesti: number on ikka alles. B unustab selle.

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




Teine peatükk teises failis

Kaks mängukassetti jagavad salvestuspesa ülesannet. 1. osa: A tähistab mängu lõppenuks. 2. osa, mis avatakse esimesena, ütleb, et tuleb 1. osa lõpetada. Pärast seda avaneb 2. osa ja B kustutab salvestuse, et 1. osa saaks uuesti mängida.

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



Üks saavutus

Sprait 8 on juba loositud. Vajuta kümme korda nuppu A. Röstsai nimetab saavutuse ja sprait 8 jääb alles, kuna see on lukustamata. Nupp B kustutab selle saavutuse kohalikust failist ja lähtestab loenduri.

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



Tuba, kuhu saab sisse astuda

256×256 mängukassett. Valge punkt on kursor. Menüü on host, join või serverite nimekiri. Host avab toa nimega walk ja alustab mängu. Join kerib aadressi (numbrid, punkt, koolon) ja B loob ühenduse. Serverite

nimekiri liigub üles ja alla nooltega ning A liitub reaga. Mängus on nooled walk спрайдil 1. Üks münt on jagatud: host liigutab seda ja selle ülesvõtmine lisab selle mängija skoori 1 ning loob järgmise. Iga спрайт, kaasa arvatud sinu oma, näitab selle skoori enda kõrval. Nurk näitab, mitu mängijat on sees, LAN-aadressi ja hosti avalikku aadressi. B lahkub: sina ja teised naasete menüüsse ning järgmine host alustab uut tuba punktisummaga 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
```



Dialoog, mis ootab

A näitab järgmist rida. Funktsioon peatub iga klahvikombinatsiooni juures, kuni vajutate uuesti klahvi A.

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



Oota pool sekundit

Sõna muutub umbes poole sekundi pärast sõnast ... sõnaks „hi“. Ekraan joonistab edasi, kuni korutiin on peatatud.

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



Ringid, mis saabuvad ükshaaval

Ülevalt tilgub sisse viis ringi, kahe kaadri kaugusel teineteisest, ja siis langevad.

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



Kaks kaadrit kellast

Sprait 1 ja sprait 2 on sama tähemärgi kaks kaadrit. Pilt vahetub neli korda sekundis.

```
cls flr time spr _draw
```

Iga funktsioon

Nurksulgudes olevad argumendid on valikulised. Tabelite ja stringide indeksid algavad 1-st, välja arvatud DATA read ja veerud, mis algavad 0-st. Värvid on paleti indeksid.

Indeks

_joonista
_init
_mp
_mp_fail
_mp_join
_mp_leave
_mp_update
_set_fps
_värskendus

abs
achclear
achdef
saavutada
saavutatud
lisada
kõik
väita
atan2
bänd
bnot
bor
btn
btnd
btnp
btnu
bxor
kaamera
lae
chr
tsirk
ringitäide
kliki
clickm
klikker
klipp
cls
kaasloomine
värvi
põhikoostis
cos
costatus
loendama
kursor
dcols
del
deli
päästa
dget
drows
dset
dstr
fget
fillp
klapp
flr
ees
fset
info
ipairs
võtme väljalülitamine
klaviatuur
keystat
keystr
rida
koormus
logi
lshr
kaart
max
memcpy
memset
mget
kesk
min
hiir
mset
muusika
musicstat
teatama
helistada
järgmiseks
nhost
njoin
nkb
nkbstr
nlan
nleave
nlist
nkuula
märkus
nmängijad
nsend
nstun
ord
ovaalne
ovaalne täidis
pakkima
paarid
sõber
palrgb
palt
pcall
piiluma
piiluma2
piiluma4
perf
pget
torkima
torkima2
torkima4
printida
printida 10
printida8
pset
rekt
rektitäit
rnd
rotl
rotr
õige
rectfill
salvestada
salvestuspesa
ekraan
kerige
vali
setmap
komplektipalett
setspr
setex
seadistus
sfx

sfxget	patt	stat
sfxhdr	lauluke	alam
sfxset	laulukomplekt	aega
sfxstat	poolitatud	trida
sget	spr	tonum
sgn	sqrt	tostr
shl	srand	lahti pakkima
näitab statistikat	sset	whoami
Shr	sspr	saagikus

Elutsükkel

_init

`_init() ... end`

Käivitub üks kord mängukasseti käivitamisel. Kood, mis asub väljaspool mis tahes konksu, lisatakse samuti _init-i.

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

_update

`_update() ... end`

Mänguloogika. Kutsutakse välja 60 Hz juures, isegi kui _draw on piiratud või host on aeglasem. Loe siit nuppe.

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

_draw

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

Joonistab ühe kaadri. Ilma argumendita või 0-ga käivitub see nii tihti, kui host seda esitab. Edasi antakse n (1..999), et piirata joonistamist n kaadriga sekundis. _update jääb 60 Hz juurde.

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

_set_fps

```
_set_fps(n)
```

Muudab mängu töötamise ajal _draw piirangut. 0 tähendab piiranguta. n on piiratud väärtusega 0..999.

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

flip

```
flip()
```

Näitab ekraani ja annab järele järgmise kaadrini. Kasuta seda ainult siis, kui joonistad ise kaadritsüklit. Ära ole hõivatud - oota pärast seda.

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

time

```
time()
```

Sekundeid pärast _init.

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

Sisend

btn

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

1, kui nuppu b hoitakse all, vastasel juhul 0. p on mängija, 0..7. Ilma argumentideta tagastab see mängija 0 bitimaski. Nupud: 0 vasakule, 1 paremale, 2 üles, 3 alla, 4 A, 5 B, 6 X, 7 Y, 8 R, 9 L, 10 select, 11 start.

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

btnd

`btnd(b, [p])`

Tõsi ainult _update kaadri puhul, kui nupp b alla vajutatakse.

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

btnp

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

Vajutamisel tõene, seejärel all hoides uuesti: 30 tiksu viivitust, seejärel iga 8 tiksu järel. Nupu puudumisel tagastab maski.

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

btnu

`btnu(b, [p])`

Tõsi, nupp vabastatakse ainult raamil.

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

mouse

`mouse()`

Tagastab hiire x ja y koordinaadid mängu pikslites.

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

clickl

`clickl([p])`

Vasak hiirenupp. clickl() on all hoitud. clickl(1) on vajutusserv.

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

clickr

`clickr([p])`

Parem hiirenupp. Sama üheargumendiline serv nagu klõpsamisel.

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

clickm

`clickm([p])`

Keskmine nupp või kerimisratta klõps.

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

scroll

`scroll()`

Hiireratta liikumine selles kaadris. Positiivne ja negatiivne olenevad ratta suunast.

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

keyon

`keyon()`

Avab nuppudega juhitava tekstivälja. Üles ja alla nooleklahvid muudavad tähte A–Z. A kirjutab vastava tähe. X kustutab. Y on tühik. B kinnitab ja sulgeb välja. Klaviatuuriga sinna ei kirjutata.

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

keyoff

`keyoff()`

Peidab tekstivälja.

```
keyoff()
```

keystr

`keystr([s])`

Tekst väljal, seejärel 1 raamil B kinnitab, seejärel muutuvad üleval ja all olevad tähed. See täht on tühi, kui väli sulgub. keystr("") tühjendab teksti.

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

keystat

```
keystat()
```

1, kui tekstiväli on avatud, vastasel juhul 0.

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

Joonistamine

screen

```
screen()
```

Praeguse mängukasseti ekraanisuurus pikslites. Tagastab laiuse ja seejärel kõrguse. See on mängu eraldusvõime, mitte redaktoriakna resolutsioon.

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

cls

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

Tühjendab ekraani. Vaikimisi värv on 0.

```
cls(1)
```

color

```
color(col)
```

Määrab joonistusvärvi, mida kasutatakse juhul, kui hilisemas väljakutses col puudub. Ilma argumendita tagastatakse praegune värv.

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

camera

`camera([x, y])`

Nihutab iga draw-käsu. camera() lähtestab väärtusele 0, 0.

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

clip

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

Piirab joonistamise ristkülikuga. Nullist erinev viies argument lõikub praeguse klipiga. clip() taastab täisekraani. Klipi registrite pilguheit on kirjutuskaitstud.

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

cursor

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

Määrab järgmise koordinaatideta prinditud objekti asukoha ja valikuliselt ka värvi. cursor() tagastab x ja y väärtused.

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

pset

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

Määrab ühe ekraanipiksli. Loeneb pikslite statistika hulka.

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

pget

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

Ekraani piksli värv x- ja y-koordinaatides.

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

print

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

Joonistab 3×5 teksti (UTF-8, kärbitud Fusion fondist). Ilma x ja y-ta prinditakse kursori juures ja liigutakse järgmisele reale.

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

log

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

Lisab tühikutega eraldatud argumendid tekstipuhvrisse. pix37_run prindib selle puhvri standardväljundisse. See ei joonista.

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

print8

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

Sama mis prinditud kujul, aga 8 px Fusion fondiga.

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

print10

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

Sama mis prinditud kujul, aga 10 px Fusion fondiga.

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

line

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

Rida x0,y0-st x1,y1-ni. Kaks argumenti jätkuvad eelmisest lõpust. line() lähtestab selle ahela.

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

circ

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

Ringi kontuur. Raadius on pikslites.

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

circfill

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

Täidetud ring.

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

oval

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

Ellipsi kontuur piirava kasti sees. Nurgad on kaasavad.

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

ovalfill

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

Täidetud ellips piiravas kastis.

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

rect

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

Ristküliku kontuur. Nurgad on kaasavad.

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

rectfill

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

Täidetud ristkülik.

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

rrect

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

Ümardatud ristküliku kontuur. x ja y on ülemine vasak nurk. w ja h on suurus pikslites; mõlemad peavad olema suuremad kui 0. r on nurga raadius ja vaikimisi on see 0. See on kinnitatud lühema külje poole.

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

rrectfill

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

Täidetud ümardatud ristkülik. Samad argumendid nagu valikul rect.

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

fillp

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

16-bitine täitemuster hiljem täidetavate kujundite jaoks. 0 on täisväärtus. fillp() tagastab praeguse mustri. Bitid läbivad 4×4 mustrit.

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

pal

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

Muudab värvi c0 värviks c1. režiim 0 on joonistuspalett, 1 kuvamispalett ja 2 täitemustri palett. pal() lähtestab.

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

palrgb

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

Määrab ühe paletikirje RGB-ks, iga kanali väärtuseks 0..255.

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

palt

```
palt([col, t])
```

Läbipaistvus. palt(col) või palt(col, true) muudab värvi col läbipaistvaks. palt(col, false) muudab selle tahkeks. palt() muudab iga värvi läbipaistmatuks.

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

setpalette

```
setpalette(n)
```

Valib paletipanga n. setpalette() tagastab praeguse panga.

```
setpalette(1)
```

tline

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

Joonistab ekraanile joone, võttes samal ajal kaardipiksleid kaardipositsioonidelt mx, my, mdx ja mdy, astades kaardinäidist mööda joont.

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

Sprite'id ja kaardid

spr

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

Joonestab спрайди n x ja y koordinaatides. w ja h on suurus pikslites, vaikimisi 8. fx ja fy pööravad horisontaalselt ja vertikaalselt, kui see on nullist erinev. tex on спрайди lehe vahekaart. rot on nurk pööretes (0..1), sama suund mis sin ja cos. 0 jätab спрайди pöörlemata.

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

sspr

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

Kopeerib спрайдilehe ristküliku ekraanile, skaleerituna dw korda dh. Vaikimisi säilitatakse lähtekoodi suurus.

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

sget

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

Ühe tekseli värv спрайdi lehel.

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

sset

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

Kirjutab спрайтilehele ühe tekseli.

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

fget

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

Sprite'i lipud. fget(n) tagastab lipubaidi. fget(n, f) on tõene, kui lipubitt f (0..7) on seatud.

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

fset

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

Määrab спрайdi lipud. fset(n, v) kirjutab terve baidi. fset(n, f, v) määrab ühe biti.

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

setspr

```
setspr(n)
```

Valib aktiivse спрайдilehe. setspr() tagastab praeguse lehe. Sama pank mis settex-il.

```
setspr(1)
```

settex

```
settex(n)
```

Setspr'i varjunimi.

```
settex(0)
```

map

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

Joonestab kaardi. Kaardi lahter cx ja cy joonistatakse ekraanil sx ja sy. cw ja ch väärtus 0 tähendab kogu kaarti. Kiht aktsepteeritakse ja ignoreeritakse. tex on paanikomplekti leht.

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

mget

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

Spraiti ID kaardi lahtris cx, cy.

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

mset

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

Kirjutab спрайди n kaardi lahtrisse cx, cy.

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

setmap

```
setmap(n)
```

Valib kaardi n. setmap() tagastab praeguse kaardi. Kaart 0 on reaalne kaart.

```
setmap(1)
```

Mälu

peek

```
peek(addr)
```

Loeb ühe baiti 64 KB aknast. Aken on reaalajas ekraan, seejärel loob спрайти piksliteks ja kaardistab lahtrid, kui ekraan mahub 65280 baiti. Suuremad ekraanid lehekülgi jagatakse käsuga poke(0x5F40, page). Kaardistamata aadressid tagastavad 0. Kirjutatakse ka @addr.

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

poke

```
poke(addr, ...)
```

Kirjutab baite aadressile addr. Lisaargumendid kirjutavad järgmised baidid. Kaardita aadresse ignoreeritakse. Kirjutatakse ka $addr, val. Ekraani puudutamine loetakse pikslikirjutuseks.

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

peek2

`peek2(addr)`

Märgita 16-bitine väärtus, little-endian, aadressil

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

poke2

`poke2(addr, val)`

Kirjutab märgita 16-bitise väärtuse, little-endian.

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

peek4

`peek4(addr)`

Märgita 32-bitine väärtus, väike-endiaan.

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

poke4

`poke4(addr, val)`

Kirjutab märgita 32-bitise väärtuse, little-endian.

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

memcpy

`memcpy(dst, src, len)`

Kopeerib len baidid samasse 64 KB aknasse.

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

memset

`memset(dest, val, len)`

Täidab len baidid alates sihtkohast väärtusega val.

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

Matemaatika

abs

`abs(x)`

Absoluutväärtus.

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

flr

`flr(x)`

Ümardab miinus lõpmatuse poole.

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

ceil

`ceil(x)`

Ümardab pluss lõpmatuse poole.

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

max

`max(a, b)`

Kahest väärtusest suurem.

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

min

`min(a, b)`

Kahest väärtusest väiksem.

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

mid

`mid(a, b, c)`

Kolmest väärtusest keskmine. Kasulik klambrina.

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

sgn

`sgn(x)`

X märk: -1, 0 või 1.

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

sqrt

`sqrt(x)`

Ruutjuur. Negatiivne argument ei ole arv.

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

sin

`sin(x)`

Siinus. x on kordamööda: 0, 0,25, 0,5, 0,75 käivad ringi üks kord. Mitte radiaanides.

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

cos

`cos(x)`

Koosinus omakorda sama kokkulepe mis siinus.

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

atan2

`atan2(dx, dy)`

Vektori dx ja dy nurk pöördetes 0-st 1-ni.

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

rnd

`rnd([x])`

Juhuslik arv vahemikus [0, x). Vaikimisi on x 1, seega rnd() on 0..1. Täringu puhul flr(rnd(6))+1.

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

srand

`srand(seed)`

Lisab juhusliku generaatori seemned, et hilisemat rnd järjestust saaks korrata.

```
srand(1)
```

band

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

Bittide kaupa JA. Kirjutatakse ka a ja b.

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

bor

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

Bitipõhine VÕI. Kirjutatakse ka a | b.

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

bxor

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

Bitipõhine XOR. Kirjutatakse ka a ^^ b.

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

bnot

```
bnot(a)
```

Bittide kaupa EI. Kirjutatakse ka ~a.

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

shl

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

Nihuta vasakule. Kirjutatakse ka << n.

shr

`shr(a, n)`

Aritmeetiline nihe paremale. Kirjutatakse ka >> n.

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

lshr

`lshr(a, n)`

Loogiline nihe paremale. Kirjutatakse ka >>> n.

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

rotl

`rotl(a, n)`

Pööra bitte vasakule. Kirjutatakse ka tähega <<> n.

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

rotr

`rotr(a, n)`

Pööra bitte paremale. Kirjutatakse ka >>< n.

Tabelid ja väärtused

add

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

Lisab tabelile t väärtuse v. Kui i on antud, lisab väärtuse sellele 1-põhisele indeksile.

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

del

`del(t, v)`

Eemaldab esimese kirje, mis on võrdne v-ga.

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

deli

`deli(t, [i])`

Eemaldab indeksi i ja tagastab selle väärtuse. Ilma i-ta eemaldab viimase kirje.

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

all

`all(t)`

Iteraator läbib massiivi osa kindlas järjekorras. Augud peatavad jalutuskäigu samamoodi nagu 1-põhine massiiv lõpeb.

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

foreach

`foreach(t, fn)`

Kutsub iga massiivi elemendi jaoks välja funktsiooni fn(v).

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

pairs

```
pairs(t)
```

Iteraator iga võtme ja väärtuse, sealhulgas nimega väljade üle.

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

ipairs

```
ipairs(t)
```

Iteraator massiivi osa üle 1-põhise indeksiga.

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

next

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

Tagastab järgmise võtme ja väärtuse pärast k. next(t) algab algusest. Kasutavad paarid.

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

count

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

Massiivi osa pikkus. Kui v on antud, loendab, mitu kirjet võrdub v-ga.

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

pack

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

Pakib argumendid uude tabelisse.

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

unpack

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

Tagastab väärtused t[i] kuni t[j] eraldi väärtustena. Vaikimisi katavad kogu massiivi osa.

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

select

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

select("#", ...) tagastab järgnevate argumentide arvu. select(i, ...) tagastab argumendid alates i-st.

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

assert

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

Peatub veaga, kui c on väär või null. msg asendab vaiketeksti.

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

pcall

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

Väljub funktsioonist f. Edu korral tagastab väärtuse „true“ ja tulemused. Vea korral tagastab väärtuse „false“ ja veateksti.

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

Keelpillid

chr

```
chr(n)
```

Üks märk baitkoodist 0..255.

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

ord

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

Indeksite i ja j baitkoodid s-dele. Indeksid on 1-põhised. Vaikimisi on algne esimene märk.

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

sub

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

Alamsõnge i-st j-ni, 1-põhine. Jäta j välja, et võtta lõpp.

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

split

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

Jagab s tabeliks. sep on vaikimisi koma. Arv sep on tüki laius. conv teisendab tükid tonum-iga, kui see on tõene.

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

tonum

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

Parsib arvu. Baas võib olla 2..36. Tagastab nulli, kui tekst ei ole arv.

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

tostr

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

Teisendab väärtuse tekstiks. digits määrab arvudele kümnendkohad.

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

Heli

sfx

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

Mängib mustrit n ja tagastab kanali. Jäta ch välja või anna edasi -1 ja valitakse esimene vaba kanal. ch -2 peatab mustri n igal kanalil. n -1 peatab kanali või iga kanali, kui ch ära jätta. n -2 laseb tsüklilisel mustril lõppeda tsükli asemel. sfx() ilma argumentideta tagastab -1.

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

sfxget

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

Tagastab ühe mustrisammu tabeli: [1] helikõrgus, [2] laine, [3] helitugevus, [4] efekt. Helikõrgus on 0..63 (C-0 kuni D#5). Laine on 0..15. Helitugevus on 0..7 ja 0 on hääletu. Efekt on 0..7.

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

sfxset

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

Kirjutab ühe mustrietapi. Kõik kuus argumenti on kohustuslikud.

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

sfxhdr

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

Mustri päis. sfxhdr(n) tagastab päisetabeli. sfxhdr(n, key, val) seab ühe välja: speed, loop_in, loop_out, level, noise, crush, dtune, rverb, dampn, swing, steps. Numbrid 0..10 valivad samad väljad selles järjekorras. speed on TIC, tikse sammu kohta. Suurem on aeglasem. 15 on 120 lööki minutis. 16-bitine aktsepteerib 1..255. Kitsamad süntesaatorid kinnituvad oma TIC-loendisse.

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

sfxstat

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

Kanalil ch esitatava mustri ID ehk -1. Vaikimisi kanal on 0.

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

note

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

Mängib ühte nooti ilma mustrita. Kõrgus 0..63, lainepikkus 0..15, helitugevus 0..7, efekt 0..7. Tikkide arv näitab, kui kaua see kestab, samades 120 Hz tikkides nagu TIC. Kui tikid välja jätta, kestab see 15, üks vaikimisi samm. Efekt jookseb selle aja jooksul, seejärel kanal vaikib.

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

music

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

Alustab lugu n ülevalt. music(-1) peatub. Funktsiooni music kutsumine ID-ga, mis juba mängib, jätab selle tööle. music() tagastab esitatava loo ID ehk -1. fade on millisekundites: lugu hakkab või hakkab hääbuma, kui n on -1. Määratud maskibitid on loo jaoks kasutatavad kanalid. 0 tähendab kõiki kaheksat. Ülejäänud kanalid jäävad sfx jaoks vabaks.

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

musicstat

```
musicstat()
```

Esitatava loo tabel. „song" ja „row" on loo ID ja rida või -1, kui midagi ei esitata. „step" ja „pos" on kaheksa numbrit. „step[1]" on kanal 0, „step[8]" on kanal 7. „step" on noodi indeks või -1, kui kanal on vaikne. „pos" näitab, kui kaugele see noot on liikunud, 0-st 1-ni. „tick" näitab, mitu tikki see rida on juba esitanud. „played" näitab, mitu rida on alustatud pärast music() funktsiooni, kaasa arvatud see, mis esitatakse.

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

songget

songget(song, [row, ch])

Mustri ID on salvestatud laululahtrisse. ch on 0..7. 255 tähendab, et lahter on tühi. See on reaalajas kutse; koodivihje tekst näitab endiselt vanemat argumentide järjekorda.

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

songset

songset(song, row, ch, pattern)

Kirjutab laululahtrisse mustri ID. Lahtri tühjendamiseks kasutage koodi 255.

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

stat

stat(n)

Ainult heli. 16..23 on kanali n-16 sfx ID. 24..31 on kanali n-24 muusikamuster. Iga muu n tagastab 0. Kaadri ajastus ja RAM on perf ja info, mitte stat.

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

Andmetabelid

dget

dget(t, row, col)

Loeb andmepaneelilt DATA andmetabeli t lahtri. Rida ja veerg algavad 0-st.

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

dset

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

Kirjutab lahtri. Kasutage tekstiveeru jaoks stringi.

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

dstr

```
dstr(t, row, col)
```

Lahter tekstina. Numbrid teisendatakse.

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

drows

```
drows(t)
```

Mitu rida on tabelis t?

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

dcols

```
dcols(t)
```

Mitu veergu on tabelis t?

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

Salvestamine ja saavutused

save

`save(k, v)`

Salvestab v salvestusfaili võtme k alla. v võib olla number, string või nende tabel. Fail on tihendatud.

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

load

`load(k)`

Loeb võtit k. Tagastab väärtuse nil, kui võti puudub.

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

delsave

`delsave(k)`

Kustutab salvestusest võtme k.

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

saveslot

`saveslot([name])`

Valib autori kaustas oleva salvestusfaili. Nime puudumisel on pesa mängu pealkiri. Tagastab väärtuse 1, kui see fail on juba olemas.

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

achieve

```
achieve(i)
```

Avab saavutuse i vahemikus 0 kuni 15. Salvestatud lokaalselt. Steami järk määrab ka ACH_i. Kataloogiseanss annab sellest teada mängija kontole.

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

achieved

```
achieved(i)
```

1 kui saavutus i on juba avatud.

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

achdef

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

Saavutuse i tervitustekst, mida kuvatakse 15 sekundit ekraani allosast. spr on valikuline 8×8 plaat, teksti selle leht.

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

achclear

```
achclear(i)
```

Tühjendab kohalikus failis oleva saavutuse i. Juba määratud Steami ACH_i jääb määratuks.

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

Võrk

whoami

`whoami()`

Sinu mängija ID on seansi ajal 0..15. Võrguühenduseta on see -1.

```
id = whoami()
```

nhost

`nhost()`

1, kui see masin on host, vastasel juhul 0.

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

nplayers

`nplayers()`

Mitu reaalajas ID-d eksisteerib vahemikus 0..15?

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

nlisten

`nlisten()`

Alustab UDP majutamist ja tagastab LAN-aadressi ip:port kujul.

```
addr = nlisten()
```

njoin

`njoin(addr)`

Liitub ip või ip:port aadressiga.

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

nleave

```
nleave()
```

Lahkub seansist. Teised näevad _mp_leave koos miks-iga 0.

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

nsend

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

Saadab kadudega sõnumi. Nimi on sündmus. Järgneb kuni neli lisaväärtust. p 0 kuni 15 sosinat sellele partnerile; jätke p välja, et saata kõigile.

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

ncall

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

Sama kuju nagu nsend, aga sõnumit proovitakse uuesti saata kuni kinnituse saamiseni. Kolm järjestikust ebaõnnestumist lõpetavad seansi.

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

nlan

```
nlan()
```

Praegune LAN ip:port pärast nlisteni.

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

nstun

```
nstun()
```

Teie avalik ip:port STUN-ist. Teises võrgus olev külaline saab proovida sellega liituda.

```
pub = nstun()
```

setturn

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

TURN varufunktsioon otsetee nurjumise korral. Tühjade argumentidega setturn() lülitab selle välja. Aadress on host:port.

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

nlist

```
nlist(game)
```

Selle mängunimega avatud seansside tabel. Igal kirjel on aadress, mängijad ja nimi.

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

nannounce

```
nannounce(game)
```

Südamelöök, et see host mängu nlistis ilmuks. Kutsu see hostimine taimeril välja.

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

nkb

`nkb([on])`

Aadressiväli. nkb(1) avab selle, nkb(0) peidab selle, nkb() näitab, kas see on avatud. Üles ja alla nooleklahvid muudavad märki, 0–9 või punkti või koolonit. A kirjutab selle. X kustutab. Klaviatuuriga sinna ei kirjutata.

```
nkb(1)
```

nkbstr

`nkbstr([s])`

Aadressiväljal olev tekst ja seejärel üleval ja all olev märk muutuvad. nkbstr("") tühjendab teksti.

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

_mp

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

Sa kirjutad selle funktsiooni. See käivitub sõnumi saabumisel. „from" on saatja ID. „name" on sündmus. Järgneb kuni neli väärtust.

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

_mp_join

`_mp_join(p)`

Sa kirjutad selle funktsiooni. Teine pool sai ID p. See käivitub ka sinu jaoks; whoami() on see ID.

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

_mp_leave

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

Sa kirjutad selle funktsiooni. Peer p lahkus. Miks 0 on tavaline lahkumine. Miks 1 on katkenud ühendus.

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

_mp_update

```
_mp_update()
```

Sa kirjutad selle funktsiooni. See töötab iga 15 _update kaadri järel, umbes 4 Hz sagedusega. Hea koht positsioonide saatmiseks.

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

_mp_fail

```
_mp_fail()
```

Sa kirjutad selle funktsiooni. Sessioon on surnud: host on kadunud või ncall ebaõnnestus kolm korda.

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

Korutiinid

cocreate

```
cocreate(f)
```

Loob funktsioonist f korutiini. See ei käivitu enne coresume'i.

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

coresume

`coresume(c, ...)`

Käivitab korutiini c, kuni see annab tulemuseks või tagastab väärtuse. Edu korral: tõene ja saadud väärtused. Vea korral: väär ja veateade.

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

costatus

`costatus(c)`

„peatatud“, „töötab“ või „surnud“.

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

yield

`yield(...)`

Annab kontrolli tagasi sellele, kes kutsus välja funktsiooni coresume. Väärtused muutuvad selle funktsiooni tulemusteks.

```
yield(1)
```

Mõõtmine

info

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

Joonistab reaalajas HUDi _draw'is: fps (1 sekundi keskmine), upd, drw, px, max, ram, peak, ramc ja net, kui seanss on avatud. info() kasutab printimiskursorit. info(col) määrab ainult värvi. info(x, y) või info(x, y, col) paigutab selle. Numbrid kasutavad B, k või m tähte, et need sobiksid väikesele ekraanile. Kutset ennast pikslites ei arvestata.

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

showstats

`showstats([on])`

Sama HUD mis info puhul, joonistatakse automaatselt 0 juures, 0 pärast _draw'd, ignoreerides kaamerat ja klippi. showstats(0) peidab selle. showstats(1) või showstats() näitab seda.

```
showstats(1)
```

perf

`perf(n)`

Üks HUD-number, et saaksid selle ise joonistada. 0 kaadrit sekundis, 1 uuendatud kaader, 2 draivi, 3 pikslit, 4 maksimum (kõige raskem kaader sel sekundil), 5 muutmälu, 6 muutmälu (muutmälu pluss pakitud mängukassett), 7 maksimaalne muutmälu alates mängukasseti algusest.

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