PIX37-käyttöohje

Editoija, sitten koodaaja. Kirjan offline-kopio osoitteessa pix37.party/docs.

Toimittaja

PIX37 DEV on editori. Täällä piirrät, kirjoitat ja kuuntelet peliä, ja sitten viet pelikasetin tai erillisen koontiversion. Se voi myös pelata pelikasettia ja avata PIX37.PARTY-tiedoston ladataksesi pelikasetteja ja käyttääksesi luetteloa. PIX37 DEV EDUCATIONAL on sama editori, ja se on ilmainen. Siinä ei ole koontiversioita, pelikasettien vientiä eikä pääsyä PIX37.PARTY-tiedostoon. PIX37 Console on soitin. Se toistaa `.p37` -pelikasetteja, ja avaat PIX37.PARTY-tiedoston ladataksesi pelikasetteja ja selataksesi luetteloa.

Projektit ja pelikasetit

`.pix37` -tiedosto on projekti: lähdekoodivälilehdet, sprite-arkit, kartat, äänet ja datataulukot. `.p37` -tiedosto on pelin pelikasetti, jota pelaaja käyttää: tiiviisti pakattua tavukoodia.

Tallenteet ja saavutukset sijaitsevat tekijäkansiossa sovellustietojen vieressä. Windowsissa se näyttää tältä `C:\\Users\\Ada\\AppData\\Roaming\\PIX37\\Mira\\Pong.p37s` tallennukselle ja `C:\\Users\\Ada\\AppData\\Roaming\\PIX37\\Mira\\Pong.ach` saavutuksille. Linuxissa samat tiedostot ovat `~/.local/share/PIX37/Mira/Pong.p37s` ja `~/.local/share/PIX37/Mira/Pong.ach` . Ada on tietokoneen tili. Mira on tekijän nimi ja tämä nimi on kansio. Pong on tallennuspaikka, jonka oletusarvo on pelin nimi. `save` kirjoittaa `.p37s` -tiedoston. Ensimmäinen avattu saavutus kirjoittaa `.ach` -tiedoston. Jos tiedostot ovat jo siellä, juoksija lukee ne, kun pelikasetti käynnistyy.

Kehittäjävalikko

Esc-näppäin avaa kehittäjävalikon. Esc-näppäin pelin ollessa käynnissä palauttaa editoriin. Vahvista Enterillä tai A-näppäimellä (oletuksena V). Takaisin-näppäin on B-näppäin (oletuksena C).

Kun projektia ei ole ladattu, valikko on: LUO PROJEKTI, LATAA PROJEKTI, RAKENNA, LATAA pelikasetti, YHTEYS PIX37.PARTYYN, ASETUKSET.

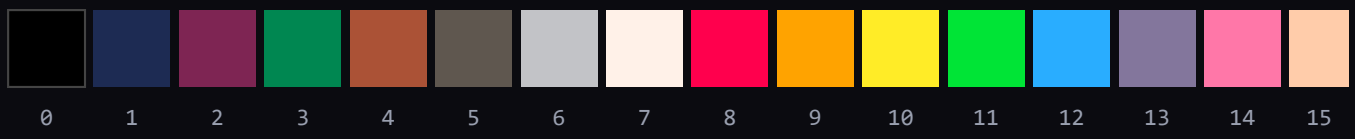
Kun projekti on ladattu, se on: TALLENNA PROJEKTI, LATAA PROJEKTI, LUO PROJEKTI, RESURSSIT, RAKENNA, LATAA pelikasetti, YHTEYS PIX37.PARTYUN, ASETUKSET.

Ctrl+S tallentaa avoimen projektin. Ctrl+R suorittaa sen.

Luo projekti

CREATE PROJECT kysyy pelikasettia ennen projektin aloittamista.

- **VIDEO** — 1-BITTI (2 väriä), 2-BITTI (4), 4-BITTI (16), 8-BITTI (256).
- **AUDIO** — syntetisaattorin näytesyvyys: 16, 8, 4, 2 tai 1 bitti.
- **NÄYTTÖ** — esiasetuksia ovat 64 × 64, 128 × 128, 160 × 90, 256 × 256 ja 320 × 180. MUKAUTETTU koko antaa sinun valita minkä tahansa koon väliltä 1 × 1 – 512 × 512.
- **PALETIT** — väripankit. Ensimmäinen pankki alkaa alla olevasta 16 värin joukosta. LOAD PALETTE käyttää sisäänrakennettua joukkoa: PICO-8 16, PICO-8 32, Game Boy, 1-bittinen, CGA, Sweetie-16 ja pidempi lista Lospecistä (Resurrect 64, Apollo, Endesga 32 ja 64, CC-29, SLSO8, Pear36, Oil 6, Duel, AAP-64, Lost Century ja muita). Voit myös ladata palettitiedoston tai ladata oman PNG-kuvasi ja ottaa paletin kyseisestä kuvasta.

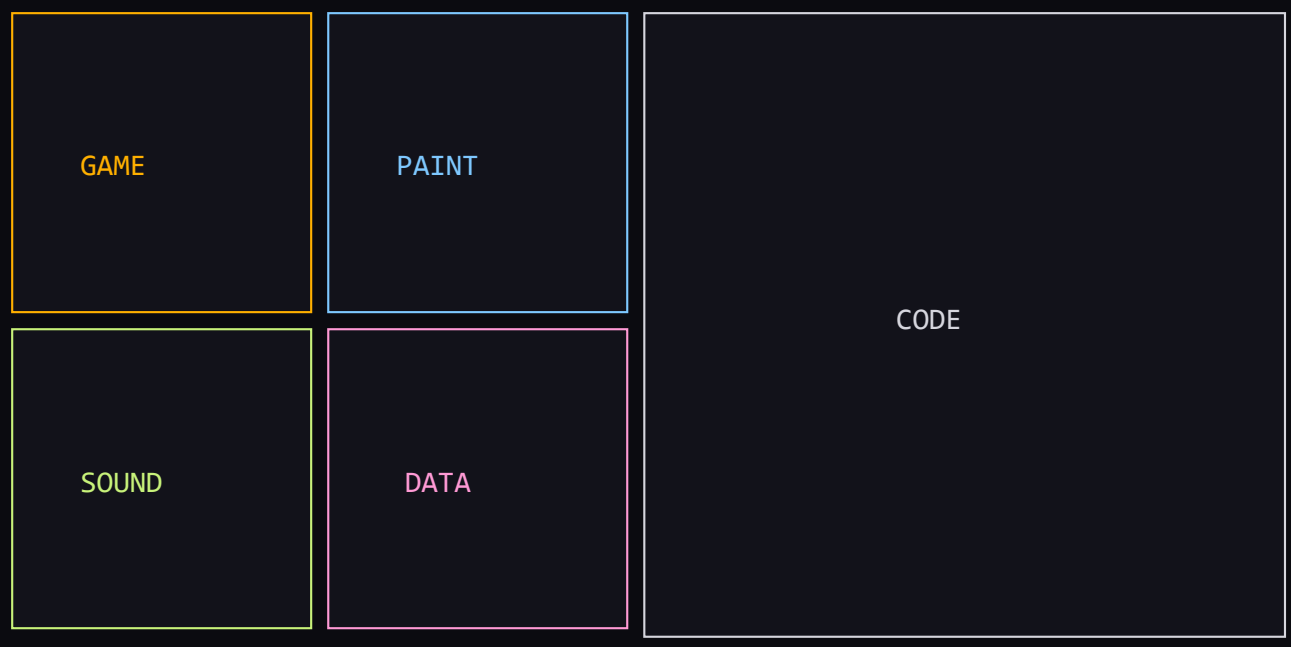



```
lette – black, navy, purple, green, brown, gray, silver, white, red, orange, yellow, lime, blue, indigo, p
```

`cls` , `pset` , `spr` ja muiden käyttämien väri-indeksien käyttö. Voit maalata minkä tahansa indeksin uudelleen palettieditorissa.

Viisi paneelia

Editori on järjestyksessä GAME, PAINT, CODE, SOUND ja DATA. Kartta sijaitsee PAINT-paneelin sisällä. SFX ja kappaleet sijaitsevat SOUND-paneelin sisällä. Napsauta paneelia työskennelläksesi siellä. Kun CODE on aktiivinen, SOUND ja DATA jättävät kirjainnäppäimet huomiotta, joten kirjoittaminen ei soita pianoa.



> Täysi asettelu, kirjoitettuna `1:1:4:1:1` . Jokainen numero osoittaa, kuinka paljon rivistä kyseinen paneeli vie: GAME, PAINT, CODE, SOUND, DATA. Pieni näyttö on keskitetty 128 × 128 -soluun. Yli 128 pikselin näyttö leventää GAME-saraketta.

ASETUKSET → YLEISET → ULKOASU vaihtaa asetuksia:

Koodi	Mitä näet
`1:1:4:1:1`	Kaikki. PELI ja ÄÄNI päällekkäin, MAALI ja DATA päällekkäin, KOODI oikealla.
`1:1:4:1:1` `2X`	Sama järjestely. KOODI piirretään kaksinkertaisella tiheydellä, joten tekstiä mahtuu kaksi kertaa enemmän.
`1:0:1:0:1`	PELI, DATA, KOODI.
`1:0:1:0:1` `2X`	Sama järjestely, tiheämmällä KOODILLA.
`0:1:1:1:1`	Ei pelin esikatselua. MAALI, KOODI, ÄÄNI, DATA.
`0:1:1:1:1` `2X`	Sama järjestely, tiheämmällä KOODILLA.
`0:1:4:1:2`	Studio. Leveä CODE, korkeampi DATA-nauha, PAINT ja SOUND.
`0:1:4:1:2` `2X`	Sama järjestely, tiheämmällä KOODILLA.
`0:0:4:0:4`	Vain KOODI ja DATA, molemmat suuria.
`0:0:4:0:4` `2X`	Sama järjestely, tiheämmällä KOODILLA.
`0:1:4:1:0`	MAALI, ÄÄNI ja laaja KOODI.
`0:1:4:1:0` `2X`	Sama järjestely, tiheämmällä KOODILLA.
`0:4:0:0:0`	Vain MAALAUS, 256×256, yksi pikseli arkista yhtä pikseliä kohden kankaasta.

`2X` rivi säilyttää samat paneelit. CODE-paneeli piirretään kaksinkertaisella resoluutiolla ja skaalataan takaisin samaan soluun, joten teksti, sarkaimet ja sisäväli ovat puolet korkeampia ja näytölle mahtuu kaksi kertaa enemmän lähdekoodia.

PELI

Käynnissä olevan pelikasetin esikatselu. Voit suorittaa esikatselun napsauttamalla PELI-paneelia tai painamalla Ctrl+R. ASETUKSET → PELI-PANEELI on paneelin konfigurointikohta:

- **Esikatselu** — tyhjä väri, jatkuva tai automaattinen uudelleenlataus muokattaessa.
- **Pelaa uudessa ikkunassa** — peli avautuu editorin viereen.
- **Näyttö** — esikatselu RGB888- tai RGB565-muodossa.
- **GAME 64** — 64×64-näyttö joko keskitetään soluun suhteessa 1:1 tai skaalataan 2× täyttämään se.
- **Varjostimet** — CRT ja pelin suodattimet, erillään editorin suodattimista.
- **Väri** — paneelin tausta.

MAALI

Sprite-arkit ja saman paneelin karttapuolella laattakartat. Arkit voivat olla suurempia kuin näyttö. Sprite-esikatselun koot ovat 8, 16, 32, 64 ja 128 (128 vain, kun arkin koko on vähintään 128 pikseliä). Lähennä sprite-esikatselua vierittämällä hiiren rullaa sen yli. Siveltimen koot ovat 1, 2, 3 ja 4.

Avain	Työkalu
B	Lyijykynä
M	Suorakulmion valinta
Q	Lasso
S	Leima
G	Ämpäri
Minä	Pipetti
V	Liikkua
L / Vaihto+L	Viiva / käyrä
U / Vaihto+U	Ontto suorakulmio / ontto ellipsi
F / Vaihto+F	Täytetty suorakulmio / täytetty ellipsi
D	Jaksaa
[ ]	Pienempi / suurempi harja
Ctrl+Z / Ctrl+Y	Kumoa / tee uudelleen. Ctrl+Shift+Z tekee myös uudelleen.
Ctrl+C / Ctrl+X / Ctrl+V	Kopioi, leikkaa, liitä
Ctrl+A	Valitse kanvas

Kartta käyttää samaa ideaa, mutta vähemmillä työkaluilla: B kynä, M suorakulmion valinta, S leima, V siirto ja Ctrl+Z, Ctrl+Y, Ctrl+C, Ctrl+V. Pidä Alt-näppäintä painettuna zoomataksesi karttatekstuuria. Siveltimen koko kartalla pysyy samana kuin maalaustyökalut.

Jokaisella spriitillä on kahdeksan lippubittiä. Paint asettaa ne spritiin; `fget` ja `fset` lukevat ne pelissä.

ASETUKSET → MAALIMUOKKAUS on paikka, jossa paneeli määritetään:

- **Paletti** — mikä pankki piirretään, sekä lataus ja tallennus.
- **Esikatselu** — RGB888 tai RGB565.
- **Siveltimen kursori** — osoitin maalattaessa.
- **Pikselin napsautus** — miten sivellin napsauttaa pikseleitä.
- **Iso sivellin** — iso sivellin napsahtaa irti vasemmasta yläkulmastaan tai koko peittoalueeltaan.
- **Sipulikuori** — muiden kehysten haamu ja mihin suuntaan tuo haamu katsoo.
- **Kartan zoomaus** — aitoja pikseleitä eli vapaa zoomaus.
- **Värit** — käyttöliittymän värit.

KOODI

Välilehdet ovat erillisiä tiedostoja projektin sisällä. Napsauta välilehden nimeä nimetäksesi sen uudelleen; Enter vahvistaa. Useat välilehdet, jotka kukin määrittelevät `_init` , `_update` tai `_draw` on yhdistetty yhteen välilehtijärjestyksessä: ensimmäinen välilehti suoritetaan ensin, viimeinen välilehti viimeisenä. Koodi, joka ei ole koukun sisällä, tulee osaksi `_init` . Pelikasettia, jossa ei ole lainkaan koukkuja, käsitellään yhtenä `_draw` .

Avain	Toiminta
Nuolet	Siirrä kursoria. Vaihto-näppäin laajentaa valintaa.
Ctrl+Vasen/Oikea	Sanan mukaan.
Home / End, Ctrl+Home / End	Rivin alku tai loppu tai sarkain.
Välilehti	Lisää kaksi välilyöntiä. Hyväksy nimi täydennysluettelossa.
Syötä	Uusi rivi, sisennetty kuten edellinen. Hyväksyy täydennyksen.
Ctrl+C / X / V	Kopioi, leikkaa, liitä.
Ctrl+Z / Ctrl+Y	Kumoa / tee uudelleen.
Ctrl+A	Valitse välilehti.
Ctrl+D	Kopioi viiva tai valinta alaspäin.
Ctrl+/	Kommentoi tai poista kommentti käyttämällä `--` .

Ctrl+F	Etsi. F3 hyppää seuraavaan osumaan. Esc sulkee palkin.
Ctrl+G	Siirry rivinumeroon.
Paeta	Sulkee ensin etsintä-, siirtymis- tai vihjeluettelon. Toinen Esc-näppäin siirtää editorin takaisin kehittäjävalikkoon.

Vihjeitä ponnahtaa esiin kirjoittaessasi funktion nimeä. Enter tai Tab lisää nimen ja avaimen `(` jos VALMIS (on käytössä asetuksissa. ASETUKSET → KOODINEDITTORI:

- **Haku** — Ctrl+F aktiivisella välilehdellä tai kaikilla välilehdillä.
- **Teksti korkea** — rajaa symbolit 5 pikselin kokoisiksi tai näytä koko symboli.
- **Merkkijonon koko** — merkkijonoliteraalien fonttikoko.
- **Kommenttifontti** — 3×5 tai nykyisen käyttöliittymäkielen Fusion-fontti.
- **Täydellinen (** — tyypittääkö täydennys sulkeet.
- **Koodivihjeet** — päällä tai pois päältä.
- **Vihjeiden kieli** — valikon kieli eli englanti.
- **Sirppiväli** – ylimääräinen siirtymä tyhjillä riveillä.
- **Vesikouru** — rivinumerot.
- **Väri** — tausta, kommentit, sarkaimet, numerot, operaattorit, avainsanat, merkkijonot, hakukoneen funktiot, omat funktiosi, muuttujat, sulkeet, sirkumfleksi, valinta, virherivi, varoitus

ÄÄNI

Paneelissa on kaksi välilehteä. SFX on yksi kuvio: lyhyt nuottisarja. SOUND on kappale: se sijoittaa nämä kuviot kahdeksalle kanavalle. TIC ilmaisee, kuinka monta tikkiä yksi askel odottaa. Kello on 120 tikkiä sekunnissa ja yksi askel on kuudestoistaosasuutti, joten 15 tikkiä on 120 lyöntiä minuutissa. Korkeampi TIC on hitaampi kuvio. Kaksi kuviota pysyy tahdissa, kun toinen TIC jakaa toisen: 15 ja 30 pysyvät, 15 ja 8 eivät.

Piano soi, kun nuotti valitaan hiirellä. Valkoiset koskettimet ovat ZXCVBNM (CDEFGAB). Mustat koskettimet ovat SDGHJ (C# D# F# G# A#). Vie hiiri minkä tahansa arvon päälle ja kirjoita numero tai vie hiiri sen päälle ja vieritä kiekkoa. Koodin sävelkorkeudet alkavat 0:sta (C-0) 63:een (D#5).

Yhdessä kuviossa LOOP ja LEN ovat samat kaksi numeroa. OUT yli 0:n toistaa silmukoita IN:stä OUT:iin, ja otsikossa lukee LOOP. OUT nollassa ja IN yli 0:n on LEN: kyseinen määrä nuotteja soitetaan alusta, sitten kuvio päättyy ja otsikossa lukee LEN. Molemmat nollassa soittavat jokaisen askeleen.

Syntetisaattorin muokattavissa oleva määrä riippuu pelikasetille valitusta bittisyvyydestä. ASETUKSET → SFX-EDITORI asettaa syvyyden, siirtyykö nuotti automaattisesti eteenpäin (pois, nuottiketju vai kenttäketju), käynnistääkö välilyönti toiston uudelleen vai vaihtaako se käynnistyksen ja pysäytyksen välillä, sekä editorin värit.

16-bittinen

Jokaisella askeleella on nuotti, äänenvoimakkuus, aalto ja efekti. Äänenvoimakkuus on 0–7. 0 on äänetön.

Aallot:

- **0 kolmio**
- **1 kallistettu saha**
- **2 sahaa**
- **3 neliö**
- **4 pulssi** — kapea pulssi
- **5 urut**
- **6 kohinaa**
- **7-vaiheinen**
- **8 PWM** — neliö, jonka leveys liikkuu
- **9 pulssia 25 %**
- **10 siniä**
- **11 metallinen kohina**
- **12 FM-kello**
- **13 nyppimistä**
- **14 supersahaa**
- **15 PWM (C64)**

Vaikutukset yhteen nuottiin:

- **0 ei yhtään**
- **1 liuku** — sävelkorkeus ja äänenvoimakkuus liukuvat seuraavaan nuottiin tässä vaiheessa
- **2 vibrato** — sävelkorkeus heiluu
- **3 drop** — sävelkorkeus laskee hiljaisuuteen, tavanomainen potku
- **4 häivytys sisään**
- **5 häivytys**
- **6 arpeggio** – kävelee tämän nuotin ja seuraavat kolme, yksi 8 tikin välein. TIC 8:lla tai sen alapuolella, joka 4 tikki.
- **7 nopeaa arpeggiota** — samat neljä nuottia, joka neljäs tikki tai joka toinen, kun TIC on 8 tai alle.

Kuvion otsikko muistiinpanojen yläpuolella:

- **TIC** — tikit askelta kohden, 1–255. 15 on 120 lyöntiä minuutissa. Kirjoita luku tai askelta sitä yhdellä.
- **LOOP IN / OUT** — kun OUT on suurempi kuin 0, pattern toistuu silmukassa IN:stä OUT:iin. Jätä OUT arvoon 0, niin otsikoksi tulee LEN: IN tarkoittaa soitettujen nuottien määrää ja 0 tarkoittaa koko patternia.
- **KOHU** — koko kuvio muuttuu kohinaksi.
- **CRUSH** — jokaisen aallon pirteä muunnelma.
- **DTURE** — 0 pois päältä, 1 kevyt kertosäe, 2 voimakas kertosäe.
- **RVERB** — jälkikaiku: 0 pois päältä, 1 lyhyt, 2 pitkä.

- **DAMPN** — alipäästösuodin: 0 pois päältä, 1 kevyt, 2 voimakas.
- **TASO** — koko kuvion äänenvoimakkuus, 0–7.
- **SWING** — laahustele. Parittomat askeleet odottavat hieman kauemmin, tasaiset askeleet hieman vähemmän.
- **VAIHEET** — kuinka monta vaihetta kuvio käyttää: 32, 24, 16 tai 12.

8-bittinen

Askelella (step) on nuotti ja aalto. Neljä aaltoa ovat neliö, kolmio, saha ja kohina. Ylätunnisteessa on TIC, LOOP IN/OUT, FX ja LEVEL. TIC on yksi seuraavista: 1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 16, 20, 24, 30, 40, 60. FX on yksi efekti koko patternille: 0 ei mitään, 1 vibrato, 2 arpeggio, 3 häivytys. LEVEL on 0–3.

4-bittinen

Askel on vain nuotti. Ylätunnisteessa on TIC, LOOP (päällä tai pois päältä), WAVE (samat neljä aaltoa) ja OCT (oktaavissa nuotit sijaitsevat). TIC on yksi seuraavista: 4, 6, 8, 10, 12, 15, 20, 30.

2-bittinen

Step on tyhjä, rumpu, osuma tai murskaava lyönti. Ylätunnisteessa on TIC ja BASE. TIC on 8, 15, 16 tai 30. BASE asettaa rummun ja osuman sijainnin.

1-bittinen

Askel on nuotti. TIC on 8, 15, 16 tai 30, samat neljä nopeutta kuin 2-bittisessä.

Kappaleet

ÄÄNI-välilehti on avoinna oleva kappale. Kahdeksan saraketta ovat kanavia. Solu on kuvion tunnus. Tyhjäksi piirretty 255 tarkoittaa, että kanava on äänetön kyseisellä rivillä. X-sarake on komento: jatka, pysäytä, silmukan alku tai silmukan loppu. Stop lopettaa kappaleen rivin jälkeen. Silmukan loppu hyppää takaisin viimeisen silmukan alkuun.

Rivi päättyy, kun vasemmanpuoleisin kuvio, jossa LOOP OUT on 0, päättyy. Oikealla pidemmät kuviot eivät pidentä riviä: pidempi kuvio katkaistaan ja lyhyempi aloittaa alusta ensimmäisestä vaiheesta, kunnes rivi päättyy. Kuvio, jossa LOOP OUT on yli 0, silmukoi LOOP IN- ja LOOP OUT-merkkien välillä eikä päätä riviä. Jos jokainen rivin kuvio silmukoi, rivi pysyy siinä.

Pikanäppäimet

Avain	Toiminta
Pianon koskettimet	Kirjoita muistiinpano hiirellä valitusta vaiheesta.

Ylös / Alas	Muuta kohdistimen alla olevaa arvoa.
Avaruus	Soita avoin SFX tai ensimmäisen rivin avoin kappale. Käynnistä/pysäytä tai käynnistä uudelleen SFX EDITOR -asetuksista.
Ctrl+Z / Ctrl+Y	Kumoa / tee uudelleen.
Ctrl+C / Ctrl+X / Ctrl+V	Kopioi, leikkaa, liitä valinta.

DATA

Nimetyt taulukot. Jokainen taulukko koostuu riveistä ja sarakkeista. Koodissa `dget` , `dset` , `dstr` , `drows` ja `dcols` käyttävät näitä taulukoita. Rivi- ja sarakenumerot alkavat koodissa nollasta.

Saraketyypit, vuorotellen otsikossa:

- **totuusarvo** — yksi bitti.
- **2** — kaksi bittiä.
- **4** — neljä bittiä.
- **8** — tavu.
- **16** — 16 bittiä id-arvoille ja suuremmille luvuille.
- **spr** — sprite-tunnus.
- **str** — teksti, piirretty järjestelmäfontilla.
- **jono8** ja **jono10** — sama teksti, piirrettynä korkeammaksi.
- **f16** ja **f32** — puolitarkkuuden liukuluku ja täysi liukuluku.

Kun osoitin on sarakeotsikon päällä, Ctrl+vieritys muuttaa kyseisen sarakkeen tyyppiä. Tavallinen vieritys liikkuu sarakkeiden yli tai rivejä alaspäin riippuen siitä, kumpi on valittu kohdassa ASETUKSET → TIETONMUOKKAAJA. Ctrl vaihtaa tätä suuntaa, minkä vuoksi Ctrl+vieritys otsikon päällä muuttaa tyyppiä, kun kiekko on asetettu sarakkeisiin.

Nuolinäppäimet liikkuvat. Enter muokkaa solua. ASETUKSET → TIETOMUOKKAUS:

- **Pyörä** – vierittää sarakkeita tai rivejä.
- **Värit** — ruudukko, otsikko ja valinta.

varat

ASSETS tuo ja vie avoimena olevan osan, ei koko projektia. Spritet käyttävät PAINT-ohjelmassa avoinna olevaa spriteä avoimella arkilla. Kartta on avoin kartta. SFX on avoin kuvio. Musiikki on avoin kappale. Data on avoin taulukko. Koodi on poikkeus: jokainen välilehti kirjoitetaan ja jokainen välilehti luetaan takaisin.

Ystävällinen	Tiedosto	Mikä liikkuu
Spritet	`.png`	Avoin sprite.
Kartta	`.p37map`	Avoin kartta.
Ääniefektit	`.p37sfx`	Avoin kuvio.
Musiikki	`.p37mus`	Avoin kappale. Pattern-numerot pysyvät ennallaan. Vie SFX erikseen, jos tarvitset näitä ääniä.
Data	`.p37dat`	Avoin pöytä.
Koodi	`.lua`	Jokainen välilehti.

Asetukset

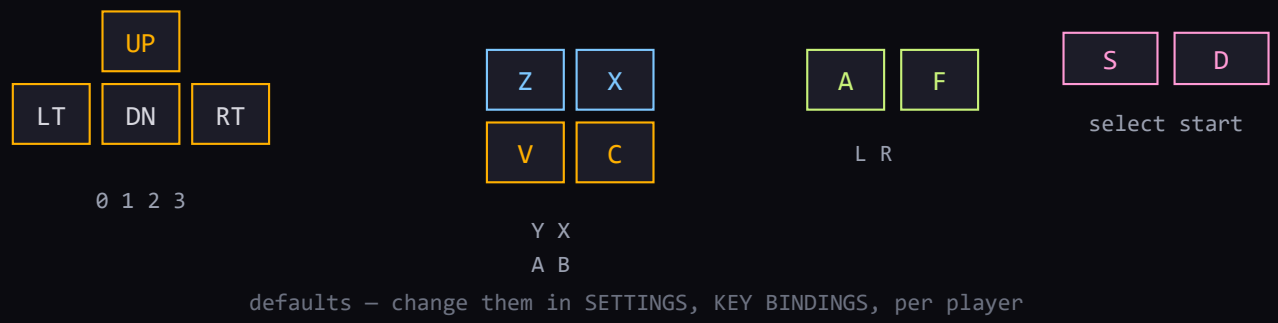
Juuriluettelo:

Sivu	Mitä se tallentaa
PROFIILIT	Tallenna ja lataa kaikki asetukset. Luo uusi, nimeä uudelleen, korvaa, poista. Kopioi tai liitä profiili tekstinä.
Valikko	Ikkuna, näppäimet, valikoiden värit, valinnainen tausta.
PELINÄYTE	Esikatselu ja toistoikkuna.
KOODIEDITORI	Korosta ja vihjeitä.
MAALIEDITORI	Sivellin, paletti, karttanäkymä.
SFX-EDITORI	Syntetisaattori ja nuottisyöttö.
TIETOJEN EDITTORI	Renkaiden suunta ja värit.
KIELI	Valikon ja vihjeiden kieli.
LISENSSIT	Lisenssitekstit, mukaan lukien Fusion Pixel -fontti ja SDL3.

MAAILMANLAAJUINEN:

- **ASENNUS** — paneelien asettelu, yllä.
- **ÄÄNENVOIMAKKUUS** — päätaso, 0–10.
- **NÄPPÄINSIDONTA** — näppäimet jokaiselle kahdeksalle pelaajalle. Paina näppäintä napataksesi sen. Ohjeet, ABXYLR, valitse.
- **Läsnä** — ohita tyhjät kehykset tai piirrä jokainen kehys.

- **Koko näyttö** — reunaton.
- **Päivitä** — ikkunan FPS. Vasen ja oikea askel 60, 90, 120 ja maksimi. Syötä luku, joka on enintään 1000. 0 tarkoittaa, ettei kuvataajuus ole rajoitettu.
- **Vsync** — odota näyttöä ruutujen välillä.
- **Skaalaus** — kokonaislukuikkunan kerroin.
- **Varjostimet** — CRT ja suodattimet editorissa, eivät pelissä.
- **Väri** — valikkoteksti ja korostus.
- **Tausta** — valinnainen PNG-kuva, rajattu 128-kokoon valikon takana. Tyhjennä-painike poistaa sen.
- **Editorin kursori** — osoittimen muoto.
- **Hiiri** — kelluva osoitin tai pikseleihin kohdistettu.
- **Käyttöliittymän nopeus** — kuinka nopeasti pidetty suunta toistuu listan läpi.
- **Resurssienhallinta** — tiedostoluettelot alkavat tietokoneen juuresta tai exe-tiedoston kansiosta.



Oletusnäppäimistö. A on vahvistus valikoissa. B on paluu. Pelissä nämä ovat painikkeet 4 ja 5. Valitse (S) on painike 10, ei valikon pikavalinta.

Painike	Rooli	Oletusavain
0	Vasen	Vasen nuoli
1	Oikea	Oikea nuoli
2	Ylös	Ylöspäin nuoli
3	Alas	Alaspäin osoittava nuoli
4	A, vahvista	V
5	B, selkä	C
6	X	X
7	Y	Z
8	R	F
9	L	A

10	Valitse	S
11	Aloita	D

Rakentaa

Valitse alusta: Windows, Linux, Mac, Android, iOS tai verkko.

- **Nimi** — koontiversion tiedostonimi.
- **Tekijä** — Windowsissa ja Linuxissa. Tämä on tallennuskansion nimi.
- **Steam-sovelluksen tunnus** — Windowsissa ja Linuxissa. Tyhjä tarkoittaa, ettei Steamworksia ole. Täytetty tunnus yhdistää `achieve(0)` … `achieve(15)` Steam-saavutuksiin ACH_0…ACH_15.
- **Id** — nipun tunniste, jos alusta pyytää sitä.
- **Piilota koodi** – tietokone- ja mobiililaitteissa. Se hidastaa pelikasetin `peek` . Se ei ole lukko.
- **Koko näyttö** — viety soitin käynnistyy koko näytön tilassa.
- **Varjostimet** – jopa kolme kerrosta, jotka on paistettu soittimeen.
- **Lataa .p37** – Vain verkossa. Näyttää latausmerkinnän taukovalikossa.
- **Kuvake** — sprite-arkin laatta: arkki, koko, x, y. Tai rajaamasi PNG-kansikuva.
- **Vie** — valitse kansio. Valikossa näkyy teksti BUILD SAVED (Kokoelma tallennettu) tai BUILD FAILED (Kokoelma epäonnistui).

Tee build internet päällä. Aloitusviesti voi kysyä palvelimelta nimettömästi, onko tämän buildin tiiviste tiedossa. Tämä tarkistus on olemassa kahdesta syystä. Se suojaa minua, PIX37:n tekijää, työkalun kopioilta, jotka eivät olleet minun, ja joita en jaa. Se suojaa myös pelin tekijää. Jonkun toisen pelin ottaminen ja sen listaaminen omaksi on yleistymässä puhelinmarkkinoilla, kuten Playssa (Android) tai iOS:ssä. Peli käynnistyy silti. Jos kone on offline-tilassa, intro sanoo, että peliä ei ole yhdistetty. Tämä rivi on myös vihje pelaajalle, että kopio ei ehkä ole peräisin laillisesta lähteestä. Se on pieni tarkistus, joten jos pelisi on varastettu ja myyty ilman suostumustasi jonkun muun toimesta, voit tehdä asiasta valituksen kaupassa, kuten Playssa (Android), Steamissa tai itch.io:ssa. Palvelin tallentaa tiivisteen ensimmäisen rekisteröintiajan tiivisteen viereen eikä muuta sitä myöhemmin. Aikaisempi aika on alkuperäinen build. Sen jälkeen rekisteröity tiiviste on myöhempi kopio.

Haluan tämän tarkistuksen suojelevan pelien tekijöitä. Pelimoottori tekee jo paljon työtä heidän puolestaan, ja tämä on yksi näistä asioista. Ymmärrän, että se voi olla kiistanalaista. Jos yhteisö päättää, että on parempi luopua tästä mikro-DRM:stä, teen niin.

Vanha pelikasetti edellisestä editorista ei välttämättä käynnisty kielenvaihdon jälkeen. Avaa `.pix37` -projekti tässä editorissa ja vie uusi `.p37` tiedosto.

Yhdistä PIX37.PARTY-palveluun

YHTEYDEN MUODOSTAMINEN PIX37.PARTY-PALVELUUHUN on sovelluksen sisällä. Selaimessa ei ole kirjautumissivua. Kirjaudu sisään Steam-istuntolipulla tai itch.io-sivustolla käyttämällä näppäimistöä ja hiirtä.

Palvelu näyttää yksilöllisen 2–16 merkkiä pitkän näyttönimen. Lisenssiavainta ja Steam-tunnusta ei näytetä muille pelaajille.

Palveluehtojen jälkeen keskiössä ovat Suosikit, Katalogi, Sijoitus, Pelaajat ja profiilisi.

Kehittäjälisenssi on editori. Sinä teet pelin, lataat sen ja jätät tähtiä. Pelaajalisenssi on pelaaja. Sydämet tulevat ihmisiltä, jotka pelaavat juoksijalla.

- **Katalogi** – selaa pelejä. Suodattimet kattavat genren (enintään kolme), näytön koon, pelikasetin koon ja sen, sisältääkö lataus `.pix37` -projektin.
- **Suosikit** — merkitsemäsi pelit.
- **Lataus** – vaatii kehittäjälisenssin. Lähetät `.p37` -tiedoston ja voit liittää mukaan `.pix37` -projektin. `.p37` on pakattua tavukoodia. Sitä on vaikea lukea, ja se voidaan tarkoituksella hämärtää. `.pix37` on projektitiedosto, jossa itse asiassa työskentelet: lähdekoodi, kuvat ja äänet muodossa, jonka toinen henkilö voi avata ja oppia siitä. Lähettämällä sen muut tekijät voivat tutkia peliä pakatun pelikasetin sijaan. Nimi, tekijä, genret, kansi, kuvake ja saavutusten nimet kulkevat sen mukana. Lähetetty RAM-muistin määrä on pelikasetin saavuttama huippu.
- **Kommentit ja saavutukset** — pelisivulla. Saavutuksen avaaminen sisäänkirjautuneena tallennetaan sekä kyseiselle tilille että pelikoneelle.
- **Sijoitus** – kehittäjät, lajiteltu tähtien tai sydänten mukaan. Jokaisella rivillä näkyvät molemmat luvut. Avaa nimi nähdäksesi kyseisen tekijän pelit sydänten mukaan järjestettyinä.
- **Pelaajat** — avattujen saavutusten mukaan järjestetyt tilit.

Koodi

Kieli on pieni Lua-tyylinen skripti. Tämä puolisko alkaa tyhjästä projektista, opettaa kielen ja näyttää sitten lyhyen esimerkin jokaisesta funktiosta.

Ensimmäinen ohjelma

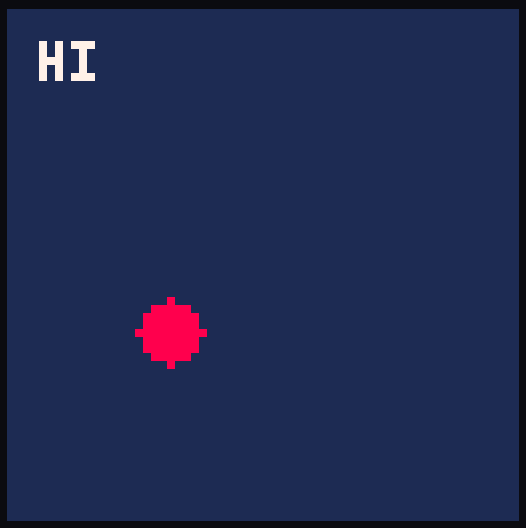
Luo projekti, jonka koko on 64×64 ja värit 4-bittiset. Avaa CODE. Korvaa välilehti seuraavalla:

```
_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 käynnistää sen. Nuolinäppäimet liikuttavat ympyrää. Esc pysäyttää pelin ja palaa editoriin.



Samat kolme kutsua: `cls(1)` piirtää tummansinisen näytön, `print` piirtää HI:n ja `circfill` piirtää vaaleanpunaisen ympyrän.

Mikä ajetaan ja milloin

- `_init` suoritetaan kerran.
- `_update` toimii 60 Hz:n taajuudella. Lisää liike ja painikkeet tähän.
- `_draw` piirtää. Ilman fps-argumenttia se piirtää niin nopeasti kuin tietokone ja näyttö pystyvät. `_draw(30)` rajoittaa piirtämisen 30 fps:ään ja jättää logiikan 60 fps:ään.
- `_set_fps(n)` muuttaa rajoitusta myöhemmin. 0 tyhjentää sen.

Nimissä ei ole kirjainkokoeroa; editori tallentaa ne pienillä kirjaimilla. `function` ja `fun` ovat sama avainsana.

Kieli

Ohjelmoinnin perusteet

Ohjelma on lista riveistä, joita pelikasetti suorittaa. Jokainen otsikko näyttää ensin viivan muodon ja sitten pienen esimerkin. Rivi, joka alkaa `--` merkillä, on kommentti: huomautus sinulle. Peli ohittaa sen.

Muuttujat

Muuttuja on nimi, jolla on arvo. Sinä valitset nimen. Myöhemmät rivit voivat lukea sen ja tallentaa uuden arvon samalla nimellä.

Muoto on `name = value` . `local name = value` pitää nimen nykyisen lohkon sisällä, joten muut pelikasetin jäsenet eivät näe sitä. Ilman `local` nimi on globaali ja jokainen välilehti voi nähdä sen.

```
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` tarkoittaa, että arvoa ei ole. `true` ja `false` ovat kyllä ja ei. Nimi, jota et ole koskaan asettanut, on `nil` .

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

Pöydät

Taulukko sisältää useita arvoja yhden nimen alla. Lista käyttää numeroita. Ensimmäinen alkio on 1, ei 0. Tietue käyttää nimiä. Taulukko voi sisältää toisen taulukon.

Listan muoto: `{ value, value }` . Tietueen muoto: `{ name = value, name = value }` . Tietueluettelo on taulukko, jonka alkiot ovat taulukoita.

```
-- 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
```

Kunto ja keho

Ehto on kysymys, johon on vastaus kyllä tai ei, kuten `hp > 0` tai `btnd(4)` . Runko on rivien joukko, joka kulkee, kun siirryt lohkoon. `if` ja jokainen silmukka käyttää samaa muotoa: ehto, sitten runko ja sitten `end` . `repeat` päättyy `until end` sijaan.

Jos

`if` suorittaa runko-osansa vain, kun ehto on tosi. `elseif` yrittää toista ehtoa. `else` suoritetaan, kun kaikki ehdot olivat epätosi.

Muoto: `if condition then` , niin kappale, niin `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
```

Silmukat

Silmukka toistaa runkoa. `while` tarkistaa ensin ehdon ja jatkaa niin kauan kuin se pysyy tosina. `repeat` suorittaa rungon ja sitten tarkistaa, että runko suoritetaan aina ainakin kerran. laskureita `for` . Viimeinen luku sisällytetään. Ilman kolmatta lukua askel on 1, joten jos alku on jo lopun jälkeen, runkoa ei suoriteta. Alaspäin laskeminen vaatii negatiivisen askeleen.

```
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
```

Pöydän äässä kävely

`all` vierailevat listan jokaisessa kohdassa ensimmäisestä alkaen. `pairs` vierailevat jokaisessa nimessä ja kyseisen nimen alla tallennetussa arvossa.

```
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
```

Toiminnot

Funktio on nimetty resepti. Muoto on `function name(arguments)` , sitten runko ja sitten `end` . Kutsut sitä kirjoittamalla nimen ja sulkeet. `return` lähettää takaisin arvon, ja se voi lähettää useamman kuin yhden. Argumentit ovat funktion lokaaleja arvoja.

```
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 funktio, joka näkee ympärillään olevat lokaalit. Se on sulkeuma. Taulukko voi sisältää sellaisen, ja sitä kutsutaan pisteellä.

```
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
```

Kyllä, ei ja valinta

`not` kääntää ehdon. `and` on tosi vain silloin, kun molemmat puolet ovat tosia. `or` on tosi, kun jompikumpi puoli on tosi. `a and b or c` valitsee `b` , kun `a` on tosi, muuten `c` . `cond ? a : b` on sama idea yhdellä rivillä: `a` kun ehto on tosi, `b` kun se 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)
```

Matematiikka ja teksti

`+ - * /` ovat yhteen-, vähennys-, kerto- ja jakolaskuja. `//` jakaa ja säilyttää kokonaisluvun. `%` on jakojäännös. `^` on potenssi. `..` yhdistää kaksi tekstiä. `#t` on listan alkioiden lukumäärä. Vertailuja ovat `==` (sama), `~=` tai `!=` (eri) ja `< > <= >=` . `x += 1` lisää 1 x:ään. Sama muoto toimii myös `-= *= /= ^= %=` .

```
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
```

Missä paikallinen asuu

`local` kestää lohkonsa sulkevan `end` asti. `if` / `end` on oma lohkonsa, voit mainita localin sen rivin yläpuolella, joka sen asettaa, mutta ennen kuin rivi on suoritettu loppuun, arvo on `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
```

Hyppääminen

`::label::` merkitsee riviä. `goto label` hyppää sinne, mutta vain saman funktion sisällä.

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

Muisti, lyhyesti sanottuna

`peek` ja `poke` katsovat eläviä pikseleitä yhden 16-bittisen osoitteen kautta. He eivät säilytä toista kopiota pelikasetista. Osoitteet `0x5F00` - `0x5FFF` ovat rekistereitä: kamera, klipsi (vain luku), piirtoväri, täyttökuvio, kolme paletin uudelleenmääritystä ja sivunumerot. Aukon molemmin puolin olevat tavut, joita on 65280, muodostavat pikseli-ikkunan.

Kun näyttö mahtuu noihin 65 280 tavuun, 64 × 64 ja 128 × 128 mahtuvat ja 256 × 256 ei, näyttö alkaa osoitteesta 0 rivi riviltä. Jäljelle jää `0x5F41` :n valitsema sprite-arkki (0 on arkki 0). Jos kyseinen arkki on suurempi kuin jäljellä oleva tila, `0x5F42` valitsee sivun eikä kartta ole ikkunassa. Jos arkki mahtuu, häntä on `0x5F43` :n valitsemat kartan solut, yksi tai kaksi tavua kutakin, ja `0x5F44` sivuttaa kartan, kun se ei mahdu häntään.

Kun näyttö ei mahdu, koko ikkuna on yksi 65 280 pikselin sivu. `poke(0x5F40, n)` kääntää näytön sivua. `poke(0x5F41, n)` jos n on muu kuin 0, näyttää arkin n, eikä arkilla 0 ole näkymää tässä tilassa. `poke(0x5F43, n)` jos n on muu kuin 0, näyttää kartan n, ja se voittaa arkin. Kartta 0 on tässä tilassa vain `mget` ja `mset` .

`peek` , joka ei osu pikseliin tai rekisteriin, palauttaa arvon 0. `poke` sinne ei tee mitään. Sprite-liput, SFX- ja DATA-taulukot eivät ole tässä ikkunassa. Käytä `fget` , `sfxget` ja `dget` .

Käytännön käyttö

Jokainen funktio on lueteltu tämän sivun alareunassa, ja mukana on PIX37-funktioiden käytännön käyttötarkoituksia. Kunkin esimerkin alla olevat linkit avaavat lisätietoja funktioista.



Sujuva liike

128×128-pelikasetti ylhäältä päin nähtynä. Sprite 1 on jo piirretty. Nuolet kiihtyvät kaikkiin suuntiin ja kitka hidastaa vauhtia. Sprite kiertää ympäri: alhaalta lähtevä puolisko piirretään ylhäälle, ja sama piirretään muille reunoille.

`_init _update _draw palt btn mid cls spr`



Tarkistetaan, onko seuraava siirto estetty

64×64-pelikasetti. Lattia ja oikeanpuoleinen seinä ovat jo kartalla, maalattu спрайtilla 3, ja lippu 0 on asetettu kyseiselle спрайtille. Nuolet liikuttavat спрайttiä 1. Yläosassa lukee "estetty", kun seuraava askel menisi päällekkäin kiinteän laatan kanssa, ja "vapaa", kun se ei menisi.

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



Pyyhkäisy käännöllä

Paina A. Pyyhkäisy piirretään flip()-funktiolla yksi musta rivi kerrallaan _draw()-funktion sijaan. _draw pysyy poissa tieltä funktion suorituksen aikana, joten jokainen rivi pysyy näytöllä.

`rectfill flip btnd cls print _update _draw`



Funktiotaulukko (luokka)

Sprite 1 on jo piirretty. make_monster toimii kuin luokka. Se palauttaa taulukon, ja update ja draw ovat metodeja, jotka säilyttävät hirviön omat x- ja y-arvot. Yläosassa oleva numero kertoo, kuinka pitkän matkan se on kävellyt.

`spr cls print flr _init _update _draw`



Neljä tapaa lukea painike

A on painike 4. Pidä sitä pohjassa, niin lataus nousee. Vapauta se, niin lataus palautuu nollaan. Ylös ja alas toistuvat painettuna, joten valikkorivi liikkuu jatkuvasti. btnd on painalluksen ensimmäinen ruutu, jota tämä pelikasetti käyttää vain pidon aloituspainikkeena.

`btn btnd btnp btnu cls print _update _draw`



Ammu hiirellä

Sprite 1 on jo piirretty keskelle. Valkoinen piste on kursori. Vasen painike lähettää pikselin sitä kohti. Rulla muuttaa nopeutta, joka on painettu kulmaan.

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

 Kirjoita nimi ja vahvista se

Avaa kenttä painamalla A. Ylös ja alas -näppäimet vaihtavat vilkkuvaa kirjainta. A kirjoittaa sen, X poistaa, Y on välilyönti ja B vahvistaa. Rivi sanoo sitten nimesi.

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

 Soittimen kamera

128×128 pelikasetti. Sprite 1 alkaa laattojen muodostaman lattian keskeltä. Nuolet liikkuvat. Kamera pitää спрайtin keskitettynä. x ja y kulmassa eivät vieritä, koska camera() ilman argumentteja palauttaa näkymän HUDia varten.

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

 Muutama rivi visuaalisesta novellista

Rivit ovat DATA-muodossa, englanniksi ja puolaksi. A jatkuu eli valinnat pysyvät näytöllä, kun vaihtoehto on näytöllä. B valitsee valinnan ja lähtee, muuten vaihtaa kieltä. X vaihtaa kieltä, kun vaihtoehto on näkyvissä. Valintaikkuna käyttää 8px-fonttia, jotta puolalaiset kirjaimet piirtyvät.

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

 Maalaa kurkistamalla ja tökkimällä

64×64-pelikasetti, jossa osoite 0 on vasemmassa yläkulmassa oleva pikseli ja seuraava rivi alkaa osoitteesta 64. Pelikasetti kirjoittaa pikseleitä, kopioi rivin, tyhjentää rivin ja tulostaa lukemansa arvot.

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

 Lista, johon voit lisätä

A lisää numeron, enintään 8:aan asti. B poistaa viimeisen. X poistaa ensimmäisen. Lista tulostuu ruudun alaosaan.

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

 Nimi purettu osiin

Pelikasetti jakaa ada,12,blue -lukujen osat ja tulostaa nimen, sen ensimmäisen kirjaimen, 12, ja sanan blue.

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

 Soita nuotti ja muuta sitä

Kuvio 0 on yhden nuotin pituinen, ja jokainen toisto on sama äänenvoimakkuus. A soittaa sen uudelleen, kun kyseinen nuotti on loppunut. B nostaa sävelkorkeutta ja soittaa kyseisen nuotin.

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

 Pelikasettiin kirjoitettu kappale

Rivillä 0 toistuu kaksi kuviota, yksi kummallakin kahdella ensimmäisellä kanavalla. Rivi pysyy nollana. kanavat 0 ja kanava 1 ovat kunkin kanavan nuotti. Pelikasetin kuvioiden, pituuden ja silmukan muokkaaminen on tapa nähdä, mitä nämä säätimet tekevät musiikkipaneelissa.

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

 Kolme hirviötä DATA-alueella

DATA-paneelissa on jo nimisarake ja hp-sarake, joissa on lima, lepakko ja silmä. A vahingoittaa limaa. Oikealla olevat numerot ovat sen hp-pisteitä.

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

 Muista pisteet

A lisää pisteen ja tallentaa sen. Lopeta ja avaa pelikasetti uudelleen: numero on edelleen paikallaan. B unohtaa sen.

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

  Toinen luku toisessa tiedostossa

Kaksi pelikasettia jakaa tallennuspaikan tehtävän. Osa 1: A merkitsee pelin päättyneeksi. Osa 2, joka avataan ensin, kehottaa viimeistelemään osan 1. Sen jälkeen osa 2 avautuu ja B tyhjentää tallennustilan, jotta osaa 1 voidaan pelata uudelleen.

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

 Yksi saavutus

Sprite 8 on jo arvottu. Paina A-näppäintä kymmenen kertaa. Paahtoleipä nimeää saavutuksen, ja sprite 8 jää, koska se on lukitsematon. B tyhjentää saavutuksen paikallisesta tiedostosta ja nollaa laskurin.

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

 Huone, johon voi kävellä sisään

256×256 pelikasetti. Valkoinen piste on kursori. Valikko on isäntä, liittyminen tai palvelinluettelo. Isäntä avaa huoneen nimeltä walk ja aloittaa pelin. Liittyminen vierittää osoitteen (numerot, piste, kaksoispiste) ja B muodostaa yhteyden. Palvelinluettelo liikkuu ylös ja alas, ja A liittyy riviin. Pelissä nuolet walk -sprite 1. Yksi kolikko jaetaan: isäntä siirtää sitä, ja sen nostaminen lisää 1 pelaajan pisteisiin ja luo seuraavan. Jokainen sprite, myös sinun, näyttää kyseisen pistemäärän vieressään. Kulmassa näkyy, kuinka monta pelaajaa on sisällä, lähiverkon osoite ja isännän julkinen osoite. B poistuu: sinä ja muut palaatte valikkoon, ja seuraava isäntä aloittaa uuden huoneen pisteillä 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`



Vuoropuhelu, joka odottaa

A näyttää seuraavan rivin. Funktio pysähtyy jokaisen rivinvaihdon kohdalla, kunnes painat A-näppäintä uudelleen.

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



Odota puoli sekuntia

Sana vaihtuu sanasta ... sanaan hi noin puolen sekunnin kuluttua. Näyttö jatkaa piirtämistä korutiinin ollessa keskeytettynä.

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



Ympyrät, jotka saapuvat yksi kerrallaan

Viisi ympyrää tippuu ylhäältä sisään kahden kehyksen välein ja sitten putoavat.

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



Kaksi ruutua kellosta

Sprite 1 ja sprite 2 ovat saman hahmon kaksi kehystä. Kuva vaihtuu neljä kertaa sekunnissa.

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

Jokainen toiminto

Hakasulkeissa olevat argumentit ovat valinnaisia. Taulukoiden ja merkkijonojen indeksit alkavat luvusta 1, paitsi DATA-rivien ja -sarakkeiden, jotka alkavat luvusta 0. Värit ovat paletti-indeksejä.

Indeksi

_draw
_init
_mp
_mp_fail
_mp_join
_mp_leave
_mp_update
_set_fps
_päivitys
abs
achclear
achdef
saavuttaa
saavutettu
lisää
kaikki
väittää
atan2
bändi
bnot
bor
btn
btnd
btnp
btnu
bxor
kamera
kattoon
chr
circ
ympyrän täyttö
clickl
clickm
clickr
leike
cls
yhteisluominen
väri
perussuu
cos
costatus
laskea
kohdistin
dcols
del
deli
delsave
dget
drows
dset
dstr
fget
fillp
kääntää
flr
foreach
fset
tiedot
ipairs
avaimet pois
keyon
keystat
keystr
riviä
kuorma
loki
lshr
kartta
max
memcpy
memset
mget
mid
min
hiiri
mset
musiikkia
musicstat
ilmoittaa
soita
seuraavaksi
nhost
njoin
nkb
nkbstr
nlan
nleave
nlist
nkuuntele
huomautus
npelaajat
nsend
nstun
ord
soikea
soikea täyttö
pakkaus
pareja
kaveri
palrgb
palt
pcall
kurkistaa
kurkistaa 2
kurkistaa 4
perf
pget
pistää
poke2
pistää 4
tulostaa
tulostaa 10
tulosta8
pset
rect
täyttö
rnd
rotl
rotr
oikein
rrectfill
tallenna
tallennuspaikka

näyttö	sfxstat	srand
rullaa	sget	sset
valitse	sgn	sspr
setmap	shl	stat
setpaletti	näyttää tilastot	sub
setspr	shr	aika
settex	syntiä	tline
asettua	kappaleet	tonum
sfx	laulusarja	tostr
sfxget	jaettu	purkaa
sfxhdr	spr	whoami
sfxset	sqrt	tuotto

Elinkaari

_init

`_init() ... end`

Suoritetaan kerran pelikasetin käynnistyessä. Myös hookin ulkopuolella oleva koodi taitetaan _init-muuttujaan.

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

_update

`_update() ... end`

Pelilogiikka. Kutsutaan 60 Hz:n taajuudella, vaikka _draw olisi rajoitettu tai isäntä on hitaampi. Lue painikkeet täältä.

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

_draw

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

Piirtää yhden kehyksen. Ilman argumenttia tai arvolla 0 se suoritetaan niin usein kuin isäntä esittää. Välitä n (1..999) piirtääksesi vain n kehystä sekunnissa. _update pysyy 60 Hz:ssä.

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

_set_fps

```
_set_fps(n)
```

Muuttaa _draw-kattoa pelin ollessa käynnissä. 0 tarkoittaa, että peli on rajaton. n on rajoitettu arvoon 0..999.

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

flip

```
flip()
```

Näyttää näytön ja siirtyy seuraavaan kehykseen asti. Käytä tätä vain, jos piirrät omaa kehyssilmukkaa. Älä odota sen jälkeen.

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

time

```
time()
```

Sekunteja _init-kohdasta lähtien.

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

Syöttö

btn

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

1, kun nappia b pidetään pohjassa, muuten 0. p on pelaaja, 0..7. Ilman argumentteja se palauttaa pelaajan 0 bittimaskin. Painikkeet: 0 vasen, 1 oikea, 2 ylös, 3 alas, 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])`

Tosi vain _update-kehyksessä, kun painike b painetaan alas.

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

btnp

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

Tosi painettaessa, sitten uudelleen pohjassa: 30 tikin viive, sitten 8 tikin välein. Ilman painiketta se palauttaa maskin.

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

btnu

`btnu(b, [p])`

Totta vain rungossa, jossa painike vapautetaan.

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

mouse

`mouse()`

Palauttaa hiiren x- ja y-koordinaatit pelin pikseleinä.

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

clickl

`clickl([p])`

Hiiren vasen painike. clickl()-funktiota pidetään pohjassa. clickl(1) on painoreuna.

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

clickr

`clickr([p])`

Hiiren oikea painike. Sama yhden argumentin reuna kuin clickl:ssä.

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

clickm

`clickm([p])`

Keskimmäinen painike tai vierityspyörän napsautus.

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

scroll

`scroll()`

Hiiren rullan liike tässä kehyksessä. Positiivinen ja negatiivinen arvo riippuvat rullan suunnasta.

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

keyon

`keyon()`

Avaa painikkeilla ohjattavan tekstikentän. Ylös ja alas -näppäimet muuttavat kirjaimia A–Z. A kirjoittaa kyseisen kirjaimen. X poistaa. Y on välilyönti. B vahvistaa ja sulkee kentän. Näppäimistöllä ei kirjoiteta siihen.

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

keyoff

`keyoff()`

Piilottaa tekstikentän.

```
keyoff()
```

keystr

`keystr([s])`

Teksti kentässä, sitten 1 kehyksessä B vahvistaa, minkä jälkeen ylös ja alas painetut kirjaimet vaihtuvat. Kirjain on tyhjä, kun kenttä sulkeutuu. keystr("") tyhjentää tekstin.

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

keystat

`keystat()`

1, kun tekstikenttä on auki, muuten 0.

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

Piirustus

screen

`screen()`

Nykyisen pelikasetin näytön koko pikseleinä. Palauttaa leveyden ja sitten korkeuden. Tämä on pelin resoluutio, ei editori-ikkunan koko.

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

cls

`cls([col])`

Tyhjentää näytön. Oletusväri on 0.

```
cls(1)
```

color

`color(col)`

Asettaa piirtovärin, jota käytetään, kun myöhemmässä kutsussa jätetään pois col. Ilman argumenttia se palauttaa nykyisen värin.

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

camera

`camera([x, y])`

Siirtää jokaisen piirtokutsun. camera() palauttaa arvon 0, 0.

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

clip

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

Rajoittaa piirtämisen suorakulmioon. Nollasta poikkeava viides argumentti leikkaa nykyisen leikkeen. clip() palauttaa koko näytön. Leikerekisterien kurkistus on vain luku -tilassa.

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

cursor

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

Asettaa mihin seuraava tuloste ilman koordinaatteja menee ja valinnaisesti värin. cursor() palauttaa x:n ja y:n.

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

pset

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

Asettaa yhden näytön pikselin. Lasketaan pikselitilastoon.

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

pget

`pget(x, y)`

Näytön pikselin väri kohdissa x ja y.

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

print

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

Piirtää 3×5 tekstiä (UTF-8, rajattu Fusion-fontista). Ilman x:ää ja y:tä se tulostuu kohdistimen kohdalle ja siirtyy seuraavalle riville.

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

log

`log(...)`

Lisää argumentit välilyönneillä erotettuina tekstipuskuriin. pix37_run tulostaa puskurin stdout-lähtöön. Se ei piirrä.

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

print8

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

Sama kuin tulostettaessa, mutta 8 px Fusion -fontilla.

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

print10

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

Sama kuin tulostettaessa, mutta 10 px Fusion-fontilla.

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

line

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

Suora kohdasta x0,y0 kohtaan x1,y1. Kaksi argumenttia jatkuu edellisestä lopusta. line() nollaa kyseisen ketjun.

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

circ

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

Ympyrän ääriviiva. Säde on pikseleinä.

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

circfill

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

Täytetty ympyrä.

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

oval

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

Ellipsin ääriviiva rajaavan laatikon sisällä. Kulmat ovat mukaan lukien.

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

ovalfill

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

Täytetty ellipsi rajaavan laatikon sisällä.

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

rect

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

Suorakulmion ääriviiva. Kulmat ovat kaikki mukana.

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

rectfill

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

Täytetty suorakulmio.

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

rrect

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

Pyöristetty suorakulmion ääriviiva. x ja y on vasen yläkulma. w ja h ovat koko pikseleinä; molempien on oltava suurempia kuin 0. r on kulman säde ja oletusarvo on 0. Se on kiinnitetty lyhyemmän sivun puolikkaaseen.

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

rrectfill

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

Täytetty pyöristetty suorakulmio. Samat argumentit kuin oikealla.

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

fillp

`fillp([p])`

16-bittinen täyttökuvio myöhemmin täytettäville muodoille. 0 on kiinteä arvo. fillp() palauttaa nykyisen kuvion. Bitit kulkevat 4×4-kuviota pitkin.

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

pal

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

Muuttaa värin c0 uudelleen väriksi c1. tila 0 on piirtopaletti, 1 näyttöpaletti ja 2 täyttökuviopaletti. pal() nollaa asetukset.

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

palrgb

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

Asettaa yhden palettimerkinnän RGB:ksi, kunkin kanavan arvoksi 0..255.

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

palt

`palt([col, t])`

Läpinäkyvyys. palt(col) tai palt(col, true) tekee väristä läpinäkyvän. palt(col, false) tekee siitä kiinteän. palt() tekee kaikista väreistä läpinäkymättömiä.

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

setpalette

`setpalette(n)`

Valitsee palettipankin n. setpalette() palauttaa nykyisen pankin.

```
setpalette(1)
```

tline

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

Piirtää viivan näytölle samalla, kun ottaa näytteitä karttapikseleistä karttapisteistä mx, my, mdx ja mdy, ja askeltaa karttanäytettä viivaa pitkin.

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

Spritet ja kartat

spr

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

Piirtää спрайтin n kohdissa x ja y. w ja h ovat koko pikseleinä, oletusarvo on 8. fx ja fy kääntävät toisiaan vaakasuunnassa ja pystysuunnassa, kun ne eivät ole nolla. tex on спрайтin arkin välilehti. rot on kulma käännöksinä (0..1), samaan suuntaan kuin sin ja cos. 0 jättää спрайтin kiertämättömäksi.

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

sspr

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

Kopioi sprite-arkin suorakulmion näytölle skaalattuna dw kertaa dh. Oletusarvot säilyttävät lähteen koon.

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

sget

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

Yhden tekselin väri sprite-arkilla.

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

sset

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

Kirjoittaa yhden texelin sprite-arkille.

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

fget

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

Sprite-liput. fget(n) palauttaa lipputavun. fget(n, f) on true, kun lippubitti f (0..7) on asetettu.

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

fset

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

Asettaa sprite-liput. fset(n, v) kirjoittaa koko tavun. fset(n, f, v) asettaa yhden bitin.

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

setspr

```
setspr(n)
```

Valitsee aktiivisen sprite-arkin. setspr() palauttaa nykyisen arkin. Sama pankki kuin settex:ssä.

```
setspr(1)
```

settex

```
settex(n)
```

setspr:n alias.

```
settex(0)
```

map

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

Piirtää kartan. Kartan solu cx ja cy piirretään näytöllä sx ja sy. Cw ja ch, arvo 0 tarkoittaa koko karttaa. Taso hyväksytään ja jätetään huomiotta. tex on laattasarjan arkki.

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

mget

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

Sprite-tunnus karttasolussa cx, cy.

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

mset

`mset(cx, cy, n)`

Kirjoittaa спрайtin n kartan soluun cx, cy.

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

setmap

`setmap(n)`

Valitsee kartan n. setmap() palauttaa nykyisen kartan. Kartta 0 on todellinen kartta.

```
setmap(1)
```

Muisti

peek

`peek(addr)`

Lukee yhden tavun 64 kt:n ikkunasta. Ikkuna on reaaliaikainen näyttö, sitten sprite-pikseleitä ja lopuksi yhdistää solut, kun näyttö mahtuu 65280 tavuun. Suuremmat näytöt sivutetaan komennolla poke(0x5F40, page). Yhdistämättömät osoitteet palauttavat arvon 0. Kirjoitetaan myös @addr.

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

poke

`poke(addr, ...)`

Kirjoittaa tavuja osoitteeseen osoite. Lisäargumentit kirjoittavat seuraavat tavut. Yhdistämättömät osoitteet jätetään huomiotta. Kirjoitetaan myös $addr, val. Näytön muuttaminen lasketaan pikselikirjoitukseksi.

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

peek2

`peek2(addr)`

Etumerkitön 16-bittinen arvo, little-endian, osoitteessa

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

poke2

`poke2(addr, val)`

Kirjoittaa etumerkitön 16-bittisen arvon, little-endian.

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

peek4

`peek4(addr)`

Etumerkitön 32-bittinen arvo, little-endian.

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

poke4

`poke4(addr, val)`

Kirjoittaa etumerkitön 32-bittisen arvon, little-endian.

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

memcpy

`memcpy(dst, src, len)`

Kopioi len-tavut saman 64 kt:n ikkunan sisälle.

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

memset

`memset(dest, val, len)`

Täyttää len-tavut alkaen dest-kohdasta arvolla val.

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

Matematiikka

abs

`abs(x)`

Absoluuttinen arvo.

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

flr

`flr(x)`

Pyöristää kohti miinus äärettömyyttä.

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

ceil

`ceil(x)`

Pyöristää kohti pistettä plus ääretön.

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

max

`max(a, b)`

Suurempi näistä kahdesta arvosta.

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

min

`min(a, b)`

Pienempi kahdesta arvosta.

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

mid

`mid(a, b, c)`

Kolmen arvon keskimmäinen. Hyödyllinen puristimena.

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

sgn

`sgn(x)`

X:n etumerkki: -1, 0 tai 1.

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

sqrt

`sqrt(x)`

Neliöjuuri. Negatiivinen argumentti ei ole luku.

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

sin

`sin(x)`

Sini. x vuorotellen: 0, 0,25, 0,5, 0,75 kiertävät ympyrän kerran. Ei radiaaneina.

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

cos

`cos(x)`

Kosini puolestaan on sama sopimus kuin sini.

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

atan2

`atan2(dx, dy)`

Vektorin dx ja dy kulma vuorotellen 0:sta 1:een.

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

rnd

`rnd([x])`

Satunnainen luku väliltä [0, x). Oletusarvo x on 1, joten rnd() on 0..1. Nopan tapauksessa flr(rnd(6))+1.

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

srand

`srand(seed)`

Syöttää satunnaisgeneraattorin, jotta myöhempi sarja voidaan toistaa.

```
srand(1)
```

band

`band(a, b)`

Bittikohtainen JA. Kirjoitetaan myös a ja b.

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

bor

`bor(a, b)`

Bittikohtainen TAI. Kirjoitetaan myös a | b.

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

bxor

`bxor(a, b)`

Bittikohtainen XOR. Kirjoitetaan myös a ^^ b.

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

bnot

`bnot(a)`

Bittikohtainen EI. Kirjoitetaan myös ~a.

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

shl

`shl(a, n)`

Siirrä vasemmalle. Kirjoitetaan myös << n.

shr

`shr(a, n)`

Aritmeettinen siirto oikealle. Kirjoitetaan myös >> n.

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

lshr

`lshr(a, n)`

Looginen siirto oikealle. Kirjoitetaan myös >>> n.

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

rotl

`rotl(a, n)`

Kierrä bittejä vasemmalle. Kirjoitetaan myös <<> n.

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

rotr

`rotr(a, n)`

Kierrä bittejä oikealle. Kirjoitetaan myös >>< n.

Taulukot ja arvot

add

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

Liittää v:n tauluun t. Jos i on annettu, lisää arvon kyseiseen 1-pohjaiseen indeksiin.

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

del

`del(t, v)`

Poistaa ensimmäisen merkinnän, joka on yhtä suuri kuin v.

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

deli

`deli(t, [i])`

Poistaa indeksin i ja palauttaa sen arvon. Ilman i:tä poistaa viimeisen merkinnän.

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

all

`all(t)`

Iteraattori käy läpi taulukon osan järjestyksessä. Reiät pysäyttävät kävelyn samalla tavalla kuin ykköspohjainen taulukko päättyy.

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

foreach

`foreach(t, fn)`

Kutsuu fn(v):tä jokaiselle taulukon alkiolle.

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

pairs

`pairs(t)`

Iteraattori jokaisen avaimen ja arvon läpi, nimetyt kentät mukaan lukien.

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

ipairs

`ipairs(t)`

Iteraattori taulukko-osan yli, jonka indeksi on 1.

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

next

`next(t, [k])`

Palauttaa seuraavan avaimen ja arvon k:n jälkeen. next(t) alkaa alusta. Käytetään pareittain.

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

count

`count(t, [v])`

Taulukon osan pituus. Jos v on annettu, laskee, kuinka monta merkintää on yhtä suuri kuin v.

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

pack

`pack(...)`

Pakkaa argumentit uuteen taulukkoon.

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

unpack

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

Palauttaa arvot t[i] - t[j] erillisinä arvoina. Oletusarvot kattavat koko taulukko-osan.

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

select

`select(i, ...)`

select("#", ...) palauttaa seuraavien argumenttien lukumäärän. select(i, ...) palauttaa argumentit i:stä eteenpäin.

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

assert

`assert(c, [msg])`

Pysähtyy virheeseen, kun c on false tai nil. msg korvaa oletustekstin.

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

pcall

`pcall(f, ...)`

Kutsuu f-funktiota. Onnistuessaan palauttaa arvon true ja tulokset. Virhetilanteessa palauttaa arvon false ja virhetekstin.

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

Jouset

chr

`chr(n)`

Yksi merkki tavukoodista 0..255.

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

ord

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

Indeksin i ja j välisten s-merkkien tavukoodit. Indeksit ovat 1-pohjaisia. Oletusarvo on ensimmäinen merkki.

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

sub

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

Osamerkkijonosta i merkkijonoon j, 1-pohjainen. Jätä j pois, jos haluat häntää.

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

split

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

Jakaa s:n taulukkoon. sep-funktion oletusarvo on pilkku. Numero sep on palan leveys. conv muuntaa osia, joiden arvo on tonum, kun se on true.

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

tonum

`tonum(s, [base])`

Jäsentää luvun. Perusluku voi olla 2..36. Palauttaa arvon nil, kun teksti ei ole luku.

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

tostr

`tostr(v, [digits])`

Muuntaa arvon tekstiksi. digits asettaa desimaalit numeroille.

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

Audio

sfx

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

Toistaa kuvion n ja palauttaa kanavan. Jos ch jätetään pois tai ch annetaan väliin, valitaan ensimmäinen vapaa kanava. ch -2 pysäyttää kuvion n jokaisella kanavalla. n -1 pysäyttää kanavan tai jokaisen kanavan, kun ch jätetään pois. n -2 antaa silmukoidun kuvion päättyä silmukan sijaan. sfx() ilman argumentteja palauttaa -1.

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

sfxget

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

Palauttaa yhden kuvioaskeleen taulukon: [1] sävelkorkeus, [2] aalto, [3] äänenvoimakkuus, [4] efekti. Sävelkorkeus on 0..63 (C-0 - D#5). Aalto on 0..15. Äänenvoimakkuus on 0..7 ja 0 on hiljainen. Efekti on 0..7.

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

sfxset

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

Kirjoittaa yhden kuvioaskeleen. Kaikki kuusi argumenttia vaaditaan.

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

sfxhdr

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

Kuvion otsikko. sfxhdr(n) palauttaa otsikkotaulukon. sfxhdr(n, key, val) asettaa yhden kentän: speed, loop_in, loop_out, level, noise, crush, dtune, rverb, dampn, swing, steps. Numerot 0..10 valitsevat samat kentät tässä järjestyksessä. nopeus on TIC, tikit askelta kohden. Korkeampi arvo on hitaampi. 15 on 120 iskua minuutissa. 16-bittinen hyväksyy 1..255. Kapeammat syntikat napsahtavat TIC-listaansa.

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

sfxstat

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

Kanavalla toistettavan kuvion tunnus tai -1. Oletuskanava on 0.

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

note

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

Soittaa yhden nuotin ilman kuviota. Korkeus 0..63, aalto 0..15, äänenvoimakkuus 0..7, efekti 0..7. Tikit ovat kestoaika, samoissa 120 Hz:n tikeissä kuin TIC. Jos tikit jätetään pois, efekti kestää 15 sekuntia, yksi oletusaskel. Efekti kestää kyseisen ajan, minkä jälkeen kanava hiljenee.

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

music

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

Aloittaa kappaleen n ylhäältä. music(-1) pysähtyy. Kun kutsutaan music-funktiota, jonka id on jo soimassa, se jää toimimaan. music() palauttaa soivan kappaleen id:n eli -1. fade on millisekunteja: kappale voimistuu tai vaienee, kun n on -1. Asetetut maskibitit ovat kappaleelle käytettyjä kanavia. 0 tarkoittaa kaikkia kahdeksaa. Muut kanavat pysyvät vapaina sfx:lle.

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

musicstat

`musicstat()`

Toistettavan kappaleen taulukko. song ja row ovat kappaleen tunnus ja rivi tai -1, kun mitään ei soiteta. step ja pos ovat kahdeksan numeroa. step[1] on kanava 0, step[8] on kanava 7. step on nuotin indeksi tai -1, kun kanava on hiljainen. pos on se, kuinka pitkälle nuotti on soitettu, nollasta yhteen. tick on se, kuinka monta tikkiä tämä rivi on jo soitettu. played on se, kuinka monta riviä on aloitettu music()-funktion jälkeen, mukaan lukien toistettava rivi.

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

songget

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

Kappalesoluun tallennettu kuviotunnus. ch on 0..7. 255 tarkoittaa, että solu on tyhjä. Tämä on reaaliaikainen kutsu; koodivihjeteksti näyttää edelleen vanhemman argumenttijärjestyksen.

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

songset

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

Kirjoittaa kuviotunnuksen kappalesoluun. Tyhjennä solu käyttämällä 255:tä.

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

stat

`stat(n)`

Vain ääni. 16..23 on kanavan n-16 sfx-tunnus. 24..31 on kanavan n-24 musiikkikuvio. Mikä tahansa muu n palauttaa arvon 0. Kehysajoitus ja RAM ovat perf- ja info-arvoja, eivät stat-arvoja.

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

Datataulukot

dget

`dget(t, row, col)`

Lukee datataulukon t solun DATA-paneelista. Rivi ja sarake alkavat nollasta.

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

dset

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

Kirjoittaa solun. Käytä merkkijonoa tekstisarakkeena.

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

dstr

`dstr(t, row, col)`

Solu tekstinä. Numerot muunnetaan.

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

drows

`drows(t)`

Kuinka monta riviä taulukossa t on?

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

dcols

`dcols(t)`

Kuinka monta saraketta taulukossa t on?

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

Tallenteet ja saavutukset

save

`save(k, v)`

Tallentaa v:n tallennustiedostoon avaimeen k. v voi olla numero, merkkijono tai näiden taulukko. Tiedosto on pakattu.

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

load

`load(k)`

Lukee avaimen k. Palauttaa nil-arvon, jos avain puuttuu.

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

delsave

`delsave(k)`

Poistaa avaimen k tallennuksesta.

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

saveslot

`saveslot([name])`

Valitsee tallennustiedoston tekijän kansiosta. Ilman nimeä paikka on pelin nimi. Palauttaa arvon 1, jos tiedosto on jo olemassa.

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

achieve

`achieve(i)`

Avaa saavutuksen i väliltä 0–15. Tallennettu paikallisesti. Steam-koontiversio asettaa myös ACH_i:n. Luetteloistunto raportoi sen pelaajan tilille.

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

achieved

`achieved(i)`

1, kun saavutus i on jo avattu.

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

achdef

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

Saavutuksen i kohoteksti, joka näkyy 15 sekunnin ajan näytön alareunasta. spr on valinnainen 8×8-laatta, tee teksti sen arkista.

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

achclear

`achclear(i)`

Tyhjentää saavutuksen i paikallisesta tiedostosta. Jo asetettu Steam ACH_i pysyy asetettuna.

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

Verkko

whoami

`whoami()`

Pelaajatunnuksesi on 0–15 pelisession aikana. Offline-tilassa se on -1.

```
id = whoami()
```

nhost

`nhost()`

1, jos tämä kone on isäntä, muuten 0.

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

nplayers

`nplayers()`

Kuinka monta live-id:tä on olemassa luvuissa 0..15?

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

nlisten

`nlisten()`

Aloittaa UDP:n isännöinnin ja palauttaa lähiverkon osoitteen ip:port-muodossa.

```
addr = nlisten()
```

njoin

`njoin(addr)`

Liittyy ip- tai ip:port-osoitteeseen.

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

nleave

`nleave()`

Poistuu istunnosta. Vertaiskäyttäjät näkevät _mp_leave-komennon, jonka syy on 0.

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

nsend

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

Lähettää häviöllisen viestin. Nimi on tapahtuma. Seuraa enintään neljä lisäarvoa. p on väliltä 0–15 kuiskausta kyseiselle vertaiskoneelle; jätä p pois, jos haluat lähettää sen kaikille.

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

ncall

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

Sama muoto kuin nsend, mutta viestiä yritetään uudelleen, kunnes se kuitataan. Kolme peräkkäistä epäonnistumista lopettaa istunnon.

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

nlan

`nlan()`

Nykyinen lähiverkon ip:portti nlistenin jälkeen.

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

nstun

`nstun()`

Julkinen IP-porttisi STUNista. Toisessa verkossa oleva vieras voi yrittää liittyä siihen.

```
pub = nstun()
```

setturn

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

TURN varamenetelmä suoran polun epäonnistuessa. setturn() tyhjillä argumenteilla poistaa sen käytöstä. Osoite on host:port.

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

nlist

`nlist(game)`

Taulukko avoimista pelisessioista kyseiselle pelinimelle. Jokaisella merkinnällä on osoite, pelaajat ja nimi.

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

nannounce

`nannounce(game)`

Sydämenlyönti, jotta tämä isäntä näkyy pelin nlistissä. Soita sille ajastimella isännöinnin aikana.

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

nkb

`nkb([on])`

Osoitekenttä. nkb(1) avaa sen, nkb(0) piilottaa sen, nkb() kertoo, onko se auki. Ylös- ja alas-näppäimet muuttavat merkkiä, 0–9, pistettä tai kaksoispistettä. A kirjoittaa sen. X poistaa. Näppäimistö ei kirjoita siihen.

```
nkb(1)
```

nkbstr

`nkbstr([s])`

Osoitekentän teksti ja sitten ylös ja alas olevat merkit muuttuvat. nkbstr("") tyhjentää tekstin.

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

_mp

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

Sinä kirjoitat tämän funktion. Se suoritetaan, kun viesti saapuu. from on lähettäjän tunniste. name on tapahtuma. Sen jälkeen voi olla enintään neljä arvoa.

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

_mp_join

`_mp_join(p)`

Kirjoitat tämän funktion. Vertaistukiyksikkö vastaanotti id:n p. Se myös laukaisee funktion sinulle; whoami() on tuo id.

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

_mp_leave

`_mp_leave(p, why)`

Kirjoitat tämän funktion. Vertaiskäyttäjä p lähti. Miksi 0 on normaali lähtö. Miksi 1 on katkennut yhteys.

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

_mp_update

`_mp_update()`

Sinä kirjoitat tämän funktion. Se suoritetaan 15 _update-kehyksen välein, noin 4 Hz:n taajuudella. Hyvä paikka nsend-sijainneille.

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

_mp_fail

`_mp_fail()`

Kirjoitat tämän funktion. Istunto on kuollut: isäntä on kadonnut tai ncall epäonnistui kolme kertaa.

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

Korutiinit

cocreate

`cocreate(f)`

Luo korutiinin funktiosta f. Se ei käynnisty ennen kuin se suorittaa coresume-komennon.

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

coresume

`coresume(c, ...)`

Suorittaa korutiinia c, kunnes se antaa tulokseksi tai palauttaa arvon. Onnistuessaan: true ja saadut arvot. Virhetilanteessa: false ja virhe.

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

costatus

`costatus(c)`

"keskeytetty", "käynnissä" tai "kuollut".

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

yield

`yield(...)`

Antaa hallinnan takaisin sille, joka kutsui coresume-funktion. Arvoista tulee kyseisen coresume-funktion tuloksia.

```
yield(1)
```

Mitata

info

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

Piirtää reaaliaikaisen HUDin _draw-funktiossa: fps (1 sekunnin keskiarvo), upd, drw, px, max, ram, peak, ramc ja net, kun istunto on käynnissä. info() käyttää tulostuskursoria. info(col) asettaa vain värin. info(x, y) tai info(x, y, col) sijoittaa sen. Numerot käyttävät B-, k- tai m-merkkejä, jotta ne sopivat pienelle näytölle. Itse kutsua ei lasketa pikselimäärään.

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

showstats

`showstats([on])`

Sama HUD kuin infossa, piirretään automaattisesti kohdassa 0, 0 _drawn jälkeen, jättäen kameran ja klipin huomiotta. showstats(0) piilottaa sen. showstats(1) tai showstats() näyttää sen.

```
showstats(1)
```

perf

`perf(n)`

Yksi HUD-numero, jotta voit piirtää sen itse. 0 fps, 1 päivitys, 2 piirtoa, 3 px, 4 max (tämän sekunnin painavin ruutu), 5 RAM-muistia, 6 RAM-muistia (RAM ja pakattu pelikasetti), 7 huippuRAM-muistia pelikasetin aloittamisesta lähtien.

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