PIX37-handleiding

Eerst de editor, dan de code. Een offline exemplaar van het boek is te vinden op pix37.party/docs.

Editor

PIX37 DEV is de editor. Hier teken, schrijf en beluister je het spel, waarna je een gamecartridge of een standalone build kunt exporteren. Je kunt er ook gamecartridges mee afspelen en PIX37.PARTY openen om gamecartridges te downloaden en de catalogus te bekijken. PIX37 DEV EDUCATIONAL is dezelfde editor, maar dan gratis. Deze versie biedt geen builds, geen mogelijkheid om gamecartridges te exporteren en geen toegang tot PIX37.PARTY. PIX37 Console is de speler. Hiermee kun je `.p37` gamecartridges afspelen en PIX37.PARTY openen om gamecartridges te downloaden en de catalogus te bekijken.

Projecten en gamecartridges

Een `.pix37`-bestand is het project: brontabbladen, sprite-sheets, kaarten, geluid en gegevenstabellen. Een `.p37`-bestand is de gamecartridge die de speler uitvoert: compact gecomprimeerde bytecode.

Opgeslagen spelbestanden en prestaties bevinden zich in de map van de auteur, naast de app-gegevens. Op Windows ziet dat eruit als `C:\\Users\\Ada\\AppData\\Roaming\\PIX37\\Mira\\Pong.p37s` voor een opgeslagen spelbestand en `C:\\Users\\Ada\\AppData\\Roaming\\PIX37\\Mira\\Pong.ach` voor prestaties. Op Linux zijn dezelfde bestanden `~/.local/share/PIX37/Mira/Pong.p37s` en `~/.local/share/PIX37/Mira/Pong.ach`. Ada is het account op de computer. Mira is de naam van de auteur, en die naam is de map. Pong is de opslaglocatie, die standaard de speltitel bevat. `save` schrijft `.p37s`. De eerste ontgrendelde prestatie schrijft `.ach`. Als de bestanden er al zijn, leest de runner ze wanneer de gamecartridge start.

Ontwikkelaarsmenu

Met Escape open je het ontwikkelaarsmenu. Als een spel draait, keer je met Escape terug naar de editor. Bevestig met Enter of de A-knop (standaard V). Teruggaan doe je met de B-knop (standaard C).

Zonder geladen project is het menu als volgt: PROJECT MAKEN, PROJECT LADEN, BOUWEN, Gamecartridge LADEN, VERBINDEN MET PIX37.PARTY, INSTELLINGEN.

Als er een project is geladen, zijn de opties: PROJECT OPSLAAN, PROJECT LADEN, PROJECT MAKEN, ACTIVA, BOUWEN, Gamecartridge LADEN, VERBINDEN MET PIX37.PARTY, INSTELLINGEN.

Met Ctrl+S sla je het geopende project op. Met Ctrl+R voer je het uit.

Een project maken

Bij het aanmaken van een project wordt om de gamecartridge gevraagd voordat je met het project kunt beginnen.

- **VIDEO** — 1-BIT (2 kleuren), 2-BIT (4), 4-BIT (16), 8-BIT (256).
- **AUDIO** — samplediepte van de synthesizer: 16, 8, 4, 2 of 1 bit.
- **SCHERM** — vooraf ingestelde formaten zijn 64×64, 128×128, 160×90, 256×256 en 320×180. Met AANGEPAST formaat kunt u elk formaat kiezen van 1×1 tot 512×512.
- **PALETTEN** — kleurenbanken. De eerste bank begint met de onderstaande set van 16 kleuren. PALET LADEN past een ingebouwde set toe: PICO-8 16, PICO-8 32, Game Boy, 1-bit, CGA, Sweetie-16 en een langere lijst van Lospec (Resurrect 64, Apollo, Endesga 32 en 64, CC-29, SLSO8, Pear36, Oil 6, Duel, AAP-64, Lost Century en andere). Je kunt ook een paletbestand laden of je eigen PNG-bestand laden en het palet uit die afbeelding halen.



Kleurindexen die worden gebruikt door `cls`, `pset`, `spr` en de rest. U kunt elke index opnieuw inkleuren in de paleteditor.

De vijf panelen

De editor bestaat uit de panelen GAME, PAINT, CODE, SOUND en DATA, in die volgorde. De kaart bevindt zich in PAINT. Geluidseffecten en muziek bevinden zich in SOUND. Klik op een paneel om daarin te werken. Wanneer CODE actief is, negeren SOUND en DATA de lettertoetsen, zodat typen niet tegelijkertijd de piano bespeelt.



Volledige lay-out, geschreven als `1:1:4:1:1`. Elk getal geeft aan hoeveel ruimte het betreffende paneel in de rij inneemt: GAME, PAINT, CODE, SOUND, DATA. Een klein scherm bevindt zich in het midden van een cel van 128×128 pixels. Een scherm groter dan 128 pixels verbreedt de GAME-kolom.

INSTELLINGEN → ALGEMEEN → INDELING wisselt tussen de verschillende indelingen:

Code	Wat je ziet
`1:1:4:1:1`	Alles. Spel en geluid boven elkaar, verf en data boven elkaar, code rechts.
`1:1:4:1:1 2X`	Dezelfde indeling. CODE wordt met een dubbele dichtheid weergegeven, waardoor er twee keer zoveel tekst op past.
`1:0:1:0:1`	SPEL, GEGEVENS, CODE.
`1:0:1:0:1 2X`	Dezelfde opstelling, maar met de dichtere CODE.
`0:1:1:1:1`	Geen spelvoorbeeld. VERF, CODE, GELUID, DATA.
`0:1:1:1:1 2X`	Dezelfde opstelling, maar met de dichtere CODE.
`0:1:4:1:2`	Studio. Brede CODE, een hogere DATA-strip, PAINT en SOUND.
`0:1:4:1:2 2X`	Dezelfde opstelling, maar met de dichtere CODE.

`0:0:4:0:4`	Alleen code en data, beide groot.
`0:0:4:0:4 2X`	Dezelfde opstelling, maar met de dichtere CODE.
`0:1:4:1:0`	VERF, GELUID en een brede CODE.
`0:1:4:1:0 2X`	Dezelfde opstelling, maar met de dichtere CODE.
`0:4:0:0:0`	Alleen verf, 256×256, één pixel van het vel per pixel van het canvas.

Een `2X`-rij behoudt dezelfde panelen. Het CODE-paneel wordt getekend met een dubbele resolutie en teruggeschaald naar dezelfde cel, waardoor de tekst, tabs en tussenruimte half zo hoog zijn en er twee keer zoveel broncode op het scherm past.

SPEL

De preview van de draaiende gamecartridge. Klik op het GAME-paneel of druk op Ctrl+R om de preview te openen. INSTELLINGEN → GAME-PANEEL is waar het paneel geconfigureerd kan worden:

- **Voorbeeldweergave** — inactieve kleur, continu of automatisch opnieuw laden tijdens het bewerken.
- **Speel in een nieuw venster** — het spel wordt naast de editor geopend.
- **Weergave** — voorbeeldweergave als RGB888 of RGB565.
- **GAME 64** — een scherm van 64×64 pixels wordt ofwel 1:1 gecentreerd in de cel, ofwel 2× vergroot om de cel te vullen.
- **Shaders** — CRT- en filterinstellingen voor het spel, los van de filters in de editor.
- **Kleur** — achtergrond van het paneel.

VERF

Sprite-sheets en, aan de kaartzijde van hetzelfde paneel, tegelkaarten. Sheets kunnen groter zijn dan het scherm. De sprite-voorbeeldformaten zijn 8, 16, 32, 64 en 128 (128 alleen als de sheet minimaal 128 px groot is). Scroll met het muiswiel over de sprite-preview om in te zoomen. De penseelgroottes zijn 1, 2, 3 en 4.

Sleutel	Hulpmiddel
B	Potlood
M	Rechthoek selecteren
Q	Lasso
S	Stempel

G	Emmer
I	Oogdruppelaar
V	Beweging
L / Shift+L	Lijn / kromme
U / Shift+U	Holle rechthoek / holle ellips
F / Shift+F	Gevulde rechthoek / gevulde ellips
D	Dither
[ ]	Kleinere / grotere kwast
Ctrl+Z / Ctrl+Y	Ongedaan maken / opnieuw uitvoeren. Ctrl+Shift+Z werkt ook als opnieuw uitvoeren.
Ctrl+C / Ctrl+X / Ctrl+V	Kopiëren, knippen, plakken
Ctrl+A	Selecteer het canvas

De kaart gebruikt hetzelfde principe, maar met minder tools: B potlood, M rechthoek selecteren, S stempel, V verplaatsen en Ctrl+Z, Ctrl+Y, Ctrl+C, Ctrl+V. Houd Alt ingedrukt om in te zoomen op de textuur van de kaart. De penseelgrootte op de kaart blijft gelijk aan die van de tekengereedschappen.

Elke sprite heeft acht vlagbits. Paint stelt deze in op de sprite; `fget` en `fset` lezen ze in het spel.

Instellingen → Tekeneditor is waar dat paneel geconfigureerd wordt:

- **Palette** — welke bank wordt gebruikt, plus laden en sparen.
- **Voorbeeldweergave** — RGB888 of RGB565.
- **De penseelcursor** is de aanwijzer waarmee je schildert.
- **Pixel snap** — hoe de kwast zich aan pixels aanpast.
- **Grote kwast** — een grote kwast breekt af vanuit de linkerbovenhoek, of vanuit zijn hele omtrek.
- **Uienschil** — een schim van andere beelden, en in welke richting die schim wijst.
- **Kaartzoom** — werkelijke pixels of een vrije zoomfunctie.
- **Kleuren** — de kleuren van de verf-UI.

CODE

Tabs zijn afzonderlijke bestanden binnen het project. Klik op een tabnaam om deze te hernoemen; druk op Enter om te bevestigen. Verschillende tabs die elk `_init`, `_update` of `_draw` definiëren, worden in tabvolgorde aan elkaar gekoppeld: de eerste tab wordt eerst uitgevoerd, de laatste als laatste. Code die zich niet binnen

een hook bevindt, wordt onderdeel van `_init`. Een gamecartridge zonder hooks wordt behandeld als één `_draw`.

Sleutel	Actie
Pijlen	Verplaats de cursor. Met Shift breid je de selectie uit.
Ctrl+Links/Rechts	Woord voor woord.
Home / End, Ctrl+Home / End	Begin of einde van de regel, of van de tab.
Tab	Voeg twee spaties in. Accepteer de naam in de invullijst.
Binnenkomen	Nieuwe regel, ingesprongen zoals de vorige. Accepteert een aanvulling.
Ctrl+C / X / V	Kopiëren, knippen, plakken.
Ctrl+Z / Ctrl+Y	Ongedaan maken / opnieuw uitvoeren.
Ctrl+A	Selecteer het tabblad.
Ctrl+D	Dupliceer de regel of de selectie naar beneden.
Ctrl+/	Reageer of verwijder de reactie met `--`.
Ctrl+F	Zoek. F3 springt naar de volgende treffer. Escape sluit de balk.
Ctrl+G	Ga naar een regelnummer.
Ontsnappen	Sluit eerst de zoekfunctie, de navigatiefunctie of de hintlijst. Een tweede keer op Escape drukken verlaat de editor en brengt je naar het ontwikkelaarsmenu.

Er verschijnen hints terwijl u een functienaam typt. Druk op Enter of Tab om de naam in te voegen, en een `(` als COMPLETE (is ingeschakeld in de instellingen. INSTELLINGEN → CODE-EDITOR:

- **Zoeken** — Ctrl+F in het actieve tabblad of in alle tabbladen.
- **Teksthoogte** — snijd de tekens bij tot 5 px, of toon het volledige teken.
- **Stringgrootte** — lettergrootte van tekenreeksliteralen.
- **Lettertype voor opmerkingen** : 3×5, of het Fusion-lettertype van de huidige UI-taal.
- **Voltooi (** — of het voltooiingstype de haakjes aangeeft.
- **Codehints** — aan of uit.
- **De taal van de hints** is de menutaal, oftewel Engels.
- **Caretspatie** — extra afstand tot lege regels.
- **Gutter** — regelnummers.
- **Kleur** — achtergrond, opmerkingen, tabs, getallen, operatoren, trefwoorden, tekenreeksen, enginefuncties, uw eigen functies, variabelen, haakjes, caret, selectie, foutlijn, waarschuwing.

GELUID

Het paneel heeft twee tabbladen. SFX is één patroon: een korte reeks noten. SOUND is een nummer: het plaatst die patronen op acht kanalen. TIC geeft aan hoeveel tikken er tussen elke stap verstrijken. De klok tikt 120 keer per seconde en één stap is een zestiende noot, dus 15 tikken is 120 slagen per minuut. Een hogere TIC betekent een langzamer patroon. Twee patronen blijven synchroon lopen wanneer de ene TIC de andere deelt: 15 en 30 wel, 15 en 8 niet.

De piano speelt wanneer een noot met de muis wordt geselecteerd. De witte toetsen zijn ZXCVBNM (CDEFGAB). De zwarte toetsen zijn SDGHJ (C# D# F# G# A#). Beweeg de muis over een waarde en typ een getal in, of beweeg de muis erover en draai aan het muiswiel. De toonhoogtes in de code lopen van 0 (C-0) tot 63 (D#5).

Bij één patroon zijn LOOP en LEN dezelfde twee getallen. OUT boven 0 herhaalt de noten van IN tot en met OUT, en de titel luidt LOOP. OUT op 0 en IN boven 0 is LEN: dat aantal noten wordt vanaf het begin gespeeld, waarna het patroon eindigt en de titel LEN luidt. Beide op 0 spelen elke stap.

Hoeveel van de synthesizer je kunt bewerken, hangt af van de bitdiepte die je voor de gamecartridge hebt gekozen. Via INSTELLINGEN → SFX-EDITOR stel je die diepte in, of een noot automatisch doorgaat (uit, notenketen of veldketen), of de spatiebalk het afspelen hervat of start en stop schakelt, en de kleuren van de editor.

16-bit

Elke stap heeft een noot, een volume, een golfvorm en een effect. Het volume loopt van 0 tot 7. 0 betekent stilte.

Golven:

- **0 driehoek**
- **1 gekantelde zaag**
- **2 zagen**
- **3 vierkant**
- **4 pulsen** — een smalle puls
- **5 organen**
- **6 geluid**
- **7 faser**
- **8 PWM** — een vierkant waarvan de breedte beweegt
- **9 pulsen 25%**
- **10 sinus**
- **11 metaalachtig geluid**
- **12 FM bel**
- **13 plukken**

- **14 supersaw**
- **15 PWM (C64)**

Effecten op één enkele noot:

- **0 geen**
- **1e slide** — toonhoogte en volume glijden naar de volgende noot over deze stap.
- **2 vibrato** — toonhoogteschommelingen
- **3 drop** — toonhoogte daalt tot stilte, de gebruikelijke kick
- **4 infade**
- **5 vervagen**
- **6 arpeggio** — deze noot en de volgende drie lopen, één om de 8 tikken. Bij TIC 8 of lager, elke 4 tikken.
- **7 snelle arpeggio** — dezelfde vier noten, elke 4 tellen, of elke 2 tellen als TIC 8 of lager is.

De patroonkop, boven de notities:

- **TIC** — ticks per stap, van 1 tot 255. 15 is 120 slagen per minuut. Typ het getal in of tel het met één op.
- **LOOP IN / OUT** — als OUT boven 0 staat, herhaalt het patroon zich van IN naar OUT. Laat OUT op 0 staan en de titel wordt LEN: IN geeft aan hoeveel noten er gespeeld worden, en 0 betekent het hele patroon.
- **RUIS** — het hele patroon wordt ruis.
- **CRUSH** — een populaire variant van elke golf.
- **DTUNE** — 0 uit, 1 een licht refrein, 2 een krachtig refrein.
- **RVERB** — nagalm: 0 uit, 1 kort, 2 lang.
- **DAMPN** — een laagdoorlaatfilter: 0 uit, 1 licht, 2 sterk.
- **NIVEAU** — volume van het hele patroon, 0 tot 7.
- **SWING** — schuifelen. Oneven stappen wachten iets langer, even stappen iets korter.
- **STAPPEN** — hoeveel stappen het patroon gebruikt: 32, 24, 16 of 12.

8-bit

Een stap bestaat uit een noot en een golf. De vier golfvormen zijn blokgolf, driehoekgolf, zaagtandgolf en ruis. De header bevat TIC, LOOP IN/OUT, FX en LEVEL. TIC is een van de waarden 1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 16, 20, 24, 30, 40, 60. FX is één effect voor het hele patroon: 0 geen, 1 vibrato, 2 arpeggio, 3 fade out. LEVEL is 0 tot 3.

4-bit

Een stap is slechts een noot. De koptekst bevat TIC, LOOP (aan of uit), WAVE (dezelfde vier golven) en OCT (in welk octaaf de noten zich bevinden). TIC is een van de volgende getallen: 4, 6, 8, 10, 12, 15, 20, 30.

2-bit

Een stap is leeg, een drum, een slag of een plet. De koptekst heeft TIC en BASE. TIC is 8, 15, 16 of 30. BASE bepaalt hoe hoog de drum en de slag zich bevinden.

1-bit

Een stap is een noot. TIC is 8, 15, 16 of 30, dezelfde vier snelheden als bij 2-bit.

Liedjes

Het tabblad GELUID is het geopende nummer. Acht kolommen zijn de kanalen. Een cel is een patroon-ID. 255, leeg weergegeven, betekent dat het kanaal op die rij stil is. De X-kolom is een commando: doorgaan, stoppen, loop starten of loop beëindigen. Stoppen beëindigt het nummer na de betreffende rij. Loop beëindigen springt terug naar de laatste loopstart.

Een rij eindigt wanneer het meest linkse patroon met LOOP OUT op 0 is voltooid. Patronen verder naar rechts verlengen die rij niet: een langer patroon wordt afgesneden, een korter patroon begint opnieuw bij de eerste stap totdat de rij eindigt. Een patroon met LOOP OUT boven 0 lus tussen LOOP IN en LOOP OUT en beëindigt de rij niet. Als elk patroon in de rij een lus vormt, blijft de rij op die plek.

Snelkoppelingen

Sleutel	Actie
Pianotoetsen	Schrijf een notitie bij de stap die met de muis is geselecteerd.
Omhoog / Omlaag	Wijzig de waarde onder de cursor.
Ruimte	Speel het openingsgeluidseffect of het openingsnummer uit de eerste rij af. Start/stop of herstart via de instellingen van de geluidseffecteneditor.
Ctrl+Z / Ctrl+Y	Ongedaan maken / opnieuw uitvoeren.
Ctrl+C / Ctrl+X / Ctrl+V	Kopieer, knip en plak de selectie.

GEGEVENS

Benoemde tabellen. Elke tabel bestaat uit rijen en kolommen. In de code gebruiken `dget`, `dset`, `dstr`, `drows` en `dcols` deze tabellen. De rij- en kolomnummers in de code beginnen bij 0.

Kolomtypen, die in de header worden weergegeven:

- **bool** — één bit.
- **2** — twee bits.

- **4** — vier bits.
- **8** — een byte.
- **16** — 16 bits, voor ID's en grotere getallen.
- **spr** — een sprite-ID.
- **str** — tekst, weergegeven met het systeemlettertype.
- **str8** en **str10** — dezelfde tekst, maar dan hoger getekend.
- **f16** en **f32** — een float met halve precisie en een float met volledige precisie.

Als de muiswijzer zich op een kolomkop bevindt, verandert Ctrl+scrollen het type van die kolom. Met gewoon scrollen ga je horizontaal of verticaal door de rijen, afhankelijk van wat je selecteert in INSTELLINGEN → GEGEVENSBEWERKER. Ctrl wisselt die richting om, vandaar dat Ctrl+scrollen op een kop het type verandert wanneer het muiswiel is ingesteld op kolommen.

Met de pijltjestoetsen kunt u navigeren. Met Enter bewerkt u een cel. INSTELLINGEN → GEGEVENSBEWERKER:

- **Het scrollwieltje** maakt het mogelijk om kolommen of rijen te scrollen.
- **Kleuren** — het raster, de koptekst en de selectie.

Activa

ASSETS importeert en exporteert het geopende onderdeel, niet het hele project. Sprites gebruikt de sprite die in PAINT is geopend, op het geopende tekenblad. De kaart is de geopende kaart. SFX is het geopende patroon. Muziek is het geopende nummer. Data is de geopende tabel. Code is de uitzondering: elk tabblad wordt geschreven en elk tabblad wordt teruggelezen.

Vriendelijk	Bestand	Wat beweegt
Elfjes	`.png`	De open sprite.
Kaart	`.p37map`	De open kaart.
SFX	`.p37sfx`	Het open patroon.
Muziek	`.p37mus`	Het openingsnummer. De patroonnummers blijven zoals ze zijn. Exporteer de geluidseffecten apart als je die geluiden nodig hebt.
Gegevens	`.p37dat`	De open tafel.
Code	`.lua`	Elk tabblad.

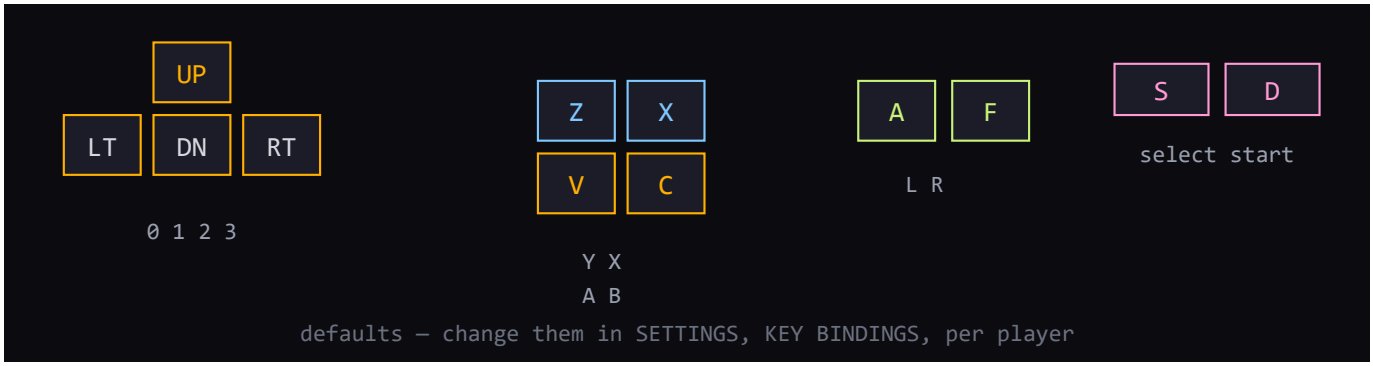
Instellingen

De hoofdlijst:

Pagina	Wat het opslaat
PROFIELEN	Alle instellingen opslaan en laden. Nieuw, hernoemen, overschrijven, verwijderen. Profiel kopiëren of plakken als tekst.
Menu	Kleuren voor venster, toetsen en menu, optionele achtergrond.
SPELPANEEL	Voorbeeldweergave en het afspeelvenster.
CODE-EDITOR	Markeringen en hints.
SCHILDERBEWERKER	Kwast, palet, kaartweergave.
SFX-EDITOR	Synthesizer en noteninvoer.
GEGEVENSBEWERKER	Richting en kleuren van de wielen.
TAAL	Menu- en hinttaal.
LICENTIES	Licentieteksten, inclusief het Fusion Pixel-lettertype en SDL3.

WERELDWIJD:

- **INDELING** — paneelindeling, hierboven.
- **VOLUME** — masterniveau, 0 tot 10.
- **TOETSENBINDINGEN** — toetsen voor elk van de 8 spelers. Druk op een toets om deze vast te leggen. Richtingen, ABXYLR, selecteren.
- **Weergeven** — inactieve frames overslaan, of elk frame tekenen.
- **Volledig scherm** — zonder randen.
- **Vernieuwen** — venster FPS. Links en rechts stappen van 60, 90, 120 en max. Druk op Enter om een getal tot 1000 in te voeren. 0 betekent geen limiet.
- **Vsync** — wacht tussen de frames op het scherm.
- **Schaal** — integer venstervermenigvuldiger.
- **Shaders** — CRT en filters in de editor, niet in het spel zelf.
- **Kleur** — menutekst en markering.
- **Achtergrond** — een optionele PNG-afbeelding, bijgesneden tot 128 pixels, achter het menu. Met 'Clear' wordt deze verwijderd.
- **Editorcursor** — aanwijzervorm.
- **Muisaanwijzer** — zwevende aanwijzer, of vastgezet op pixels.
- **UI-snelheid** — hoe snel een ingedrukte muisknop wordt herhaald in een lijst.
- In **Verkenner** beginnen bestandslijsten bij de hoofdmap van de computer, of bij de map van het uitvoerbare bestand (exe).



Standaard toetsenbord. A is bevestigen in menu's. B is terug. In een spel zijn dit de toetsen 4 en 5. Selecteren (S) is toets 10, geen menusneltoets.

Knop	Rol	Standaardsleutel
0	Links	Linkerpijl
1	Rechts	Rechterpijl
2	Omhoog	Pijl omhoog
3	Omlaag	Pijl naar beneden
4	A, bevestigen	V
5	B, achterkant	C
6	X	X
7	Y	Z
8	R	F
9	L	A
10	Select	S
11	Begin	D

Bouwen

Kies een platform: Windows, Linux, Mac, Android, iOS of web.

- **Naam** — bestandsnaam van de build.
- **Auteur** — op Windows en Linux. Dit is de mapnaam voor opgeslagen bestanden.
- **Steam-app-ID** — op Windows en Linux. Een lege ID betekent geen Steamworks-app. Een ingevulde ID koppelt `achieve(0)`…`achieve(15)` aan Steam-prestaties ACH_0…ACH_15.

- **Id** — bundel-ID indien het platform daarom vraagt.
- **Code verbergen** — op desktop- en mobiele overlays. Het vertraagt `peek` van de gamecartridge. Het is geen vergrendeling.
- **Volledig scherm** — de geëxporteerde speler start in volledig scherm.
- **Shaders** — tot wel drie lagen die in de speler zijn ingebakken.
- **Download .p37** — Alleen via internet. Toont een downloadoptie in het pauzemenu.
- **Pictogram** — een tegel uit een sprite-sheet: sheet, grootte, x, y. Of een PNG-cover die je bijsnijdt.
- **Exporteren** — selecteer de map. In het menu verschijnt 'BUILD SAVED' of 'BUILD FAILED'.

Maak een build met een internetverbinding. Het opstartscherm kan de server anoniem vragen of de hash van deze build bekend is. Deze controle dient twee doelen. Het beschermt mij, de maker van PIX37, tegen kopieën van de tool die niet van mij zijn en die ik niet mag weggeven. Het beschermt ook de maker van het spel. Het is steeds vaker voorkomend op de markt voor mobiele apparaten, zoals Google Play (Android) of iOS, om andermans spellen als je eigen spellen te presenteren. Het spel start nog steeds. Als de computer offline is, geeft de intro aan dat het spel niet verbonden is. Deze melding geeft de speler ook een hint dat de kopie mogelijk niet van een legale bron afkomstig is. Het is een kleine controle, zodat je, als je spel zonder jouw toestemming is gestolen en verkocht door iemand anders, een klacht kunt indienen bij een appwinkel zoals Google Play (Android), Steam of itch.io. De server slaat het tijdstip waarop de hash voor het eerst is geregistreerd op, naast de hash zelf, en wijzigt dit later niet. De eerdere tijd is de originele build. Een hash die daarna is geregistreerd, is een latere kopie.

Ik wil deze controle om de mensen die games maken te beschermen. De engine doet al veel werk voor hen, en dit is daar een voorbeeld van. Ik begrijp dat het controversieel kan zijn. Als de community besluit dat het beter is om deze micro-DRM te laten vallen, zal ik dat doen.

Een oude gamecartridge van een eerdere editor kan na het wijzigen van de taal weigeren op te starten. Open het project `.pix37` in deze editor en exporteer een nieuw project `.p37`.

Maak verbinding met PIX37.PARTY

De verbinding met PIX37.PARTY bevindt zich in de app. Er is geen inlogpagina in de browser. Je meldt je aan met een Steam-sessieticket of met itch.io, met behulp van de controller en de muis. De service toont een unieke weergavenaam van 2 tot 16 tekens. De licentiesleutel en de Steam-ID worden niet aan andere spelers getoond.

Na de gebruiksvoorwaarden vind je in het centrale gedeelte de volgende opties: Favorieten, Catalogus, Ranglijst, Spelers en je profiel.

Een ontwikkelaarslicentie is voor de editor. Je maakt het spel, je uploadt het en je geeft sterren. Een gamerslicentie is voor de speler. Hartjes krijg je van mensen die een runner spelen.

- **Catalogus** — blader door games. Filters omvatten genre (maximaal drie), schermgrootte, grootte van de gamecartridge en of de upload een `.pix37`-project bevat.

- **Favorieten** — spellen die je hebt gemarkeerd.
- **Uploaden** vereist een ontwikkelaarslicentie. Je stuurt een `.p37`-bestand en je kunt het `.pix37`-project bijvoegen. Een `.p37` is gecomprimeerde bytecode. Deze is moeilijk leesbaar en kan opzettelijk geobfusceerd zijn. Een `.pix37` is het projectbestand waarmee je daadwerkelijk werkt: de broncode, de afbeeldingen en het geluid, in een vorm die iemand anders kan openen en waarvan diegene kan leren. Door dit bestand te versturen, kunnen andere makers het spel bestuderen in plaats van een gecomprimeerde gamecartridge. Titel, auteur, genres, cover, icoon en namen van achievements worden meegestuurd. Het verzonden RAM-nummer geeft de maximale hoeveelheid RAM aan die de gamecartridge heeft bereikt.
- **Opmerkingen en prestaties** — op de spelpagina. Het behalen van een prestatie terwijl je bent ingelogd, wordt zowel op dat account als op de computer opgeslagen.
- **Ranglijst** — ontwikkelaars, gesorteerd op sterren of harten. Elke rij toont beide aantallen. Klik op een naam om de spellen van die auteur te bekijken, gesorteerd op harten.
- **Spelers** — accounts gesorteerd op behaalde prestaties.

Code

De taal is een klein script in Lua-stijl. Dit deel begint met een leeg project, leert de taal aan en laat vervolgens een kort voorbeeld zien van elke functie.

Een eerste programma

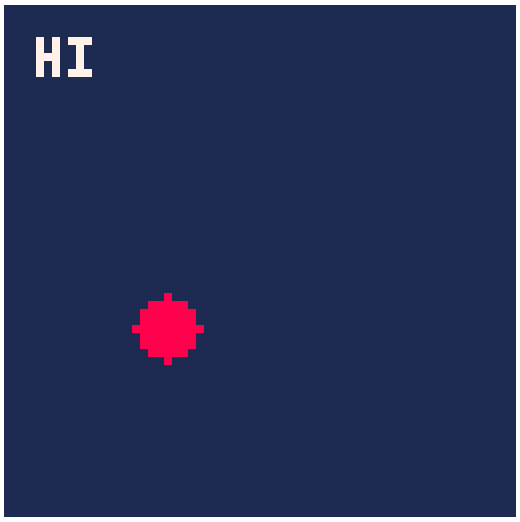
Maak een project aan met een resolutie van 64x64 pixels en 4-bits kleurdiepte. Open CODE. Vervang het tabblad door:

```
_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 start het programma. Met de pijltjestoetsen beweeg je de cirkel. Escape stopt het spel en brengt je terug naar de editor.



Dezelfde drie oproepen: `cls(1)` kleurt het donkerblauwe scherm, `print` tekent HI en `circfill` tekent de roze cirkel.

Wat loopt er, en wanneer?

- `_init` wordt eenmaal uitgevoerd.
- `_update` werkt op 60 Hz. Plaats hier beweging en knoppen.
- `_draw` tekent. Zonder argument voor de framesnelheid tekent het zo snel als de computer en de monitor aankunnen. `_draw(30)` beperkt het tekenen tot 30 fps en laat de logica op 60 staan.
- `_set_fps(n)` wijzigt die limiet later. 0 wist hem.

Namen zijn niet hoofdlettergevoelig; de editor slaat ze op in kleine letters. `function` en `fun` zijn hetzelfde trefwoord.

Taal

Basisprincipes van programmeren

Een programma is een lijst met regels die de gamecartridge uitvoert. Elke kop toont eerst de structuur van de regel, gevolgd door een klein voorbeeld. Een regel die begint met `--` is een commentaar: een notitie voor jou. Het spel slaat deze over.

Variabelen

Een variabele is een naam die een waarde bevat. Je kiest zelf de naam. Latere regels kunnen die naam lezen en er een nieuwe waarde in opslaan onder dezelfde naam.

De vorm is `name = value`. `local name = value` zorgt ervoor dat die naam binnen het huidige blok blijft, zodat de rest van de gamecartridge deze niet kan zien. Zonder `local` is de naam globaal en kan elk tabblad deze zien.

```
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` betekent dat er geen waarde is. `true` en `false` betekenen respectievelijk 'ja' en 'nee'. Een naam die u nooit hebt ingesteld, wordt weergegeven als `nil`.

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

Tabellen

Een tabel bevat meerdere waarden onder één naam. Een lijst gebruikt getallen. Het eerste item is 1, niet 0. Een record gebruikt namen. Een tabel kan een andere tabel bevatten.

Lijstvorm: `{ value, value }`. Recordvorm: `{ name = value, name = value }`. Een lijst met records is een tabel waarvan de items tabellen zijn.

```
-- 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
```

Conditie en lichaam

Een voorwaarde is een vraag met een ja- of nee-antwoord, zoals `hp > 0` of `btnd(4)`. Een body is de groep regels die worden uitgevoerd wanneer je een blok binnengaat. `if` en elke lus gebruiken dezelfde structuur: een voorwaarde, dan een body, dan `end`. `repeat` wordt afgesloten met `until` in plaats van `end`.

Als

`if` voert de code alleen uit als aan de voorwaarde is voldaan. `elseif` probeert een andere voorwaarde. `else` wordt uitgevoerd wanneer alle voorwaarden onwaar waren.

Vorm: `if condition then`, vervolgens het lichaam, dan `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
```

Lussen

Een lus herhaalt een body. `while` controleert eerst de voorwaarde en gaat door zolang deze waar blijft. `repeat` voert de body uit en controleert vervolgens, zodat de body altijd minstens één keer wordt uitgevoerd. `for` telt. Het laatste getal wordt meegeteld. Zonder een derde getal is de stap 1, dus als het begin al voorbij het einde is, wordt de body niet uitgevoerd. Aftellen vereist een negatieve stap.

```
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
```

Een tafel lopen

`all` doorloopt elk item van een lijst, beginnend bij het eerste. `pairs` doorloopt elke naam en de waarde die onder die naam is opgeslagen.

```
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
```

Functies

Een functie is een benoemd recept. De structuur is `function name(arguments)`, gevolgd door een body, en dan `end`. Je roept de functie aan door de naam en haakjes te schrijven. `return` stuurt een waarde terug, en kan er meer dan één terugsturen. De argumenten zijn lokale variabelen van die functie.

```
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` is een functie die de lokale variabelen eromheen kan zien. Dat is een closure. Een tabel kan er één bevatten en je roept hem aan met een punt.

```
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
```

Ja, nee en kiezen

`not` draait een voorwaarde om. `and` is alleen waar als beide kanten waar zijn. `or` is waar als een van beide kanten waar is. `a and b or c` kiest `b` als `a` waar is, anders `c`. `cond ? a : b` is hetzelfde idee in één regel: `a` als de voorwaarde waar is, `b` als deze niet waar is.

```
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)
```

Wiskunde en tekst

`+ - * /` staat voor optellen, aftrekken, vermenigvuldigen en delen. `//` deelt en behoudt het hele getal. `%` is de rest. `^` is machtsverheffen. `..` plakt twee teksten aan elkaar. `#t` geeft aan hoeveel items er in een lijst staan. Vergelijkingen zijn `==` (gelijk), `~=` of `!=` (verschillend) en `< > <= >=`. `x += 1` telt 1 op bij x. Dezelfde vorm werkt voor `-= *= /= ^= %=`.

```
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
```

Waar een lokale bewoner woont

`local` blijft van kracht tot `end` het bijbehorende blok afsluit. `if` / `end` is een eigen blok. Je kunt een lokale variabele boven de regel die deze waarde instelt, opgeven, maar totdat die regel wordt uitgevoerd, is de waarde `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
```

Springen

`::label::` markeert een regel. `goto label` springt daarheen, maar alleen binnen dezelfde functie.

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

Geheugen, in het kort.

`peek` en `poke` bekijken live pixels via één 16-bits adres. Ze bewaren geen tweede kopie van de gamecartridge. De adressen `0x5F00` tot en met `0x5FFF` zijn registers: camera, clip (alleen-lezen), tekenkleur, vulpatroon, de drie paletremappings en de paginanummers. De bytes aan weerszijden van dat gat, 65280 in totaal, vormen het pixelvenster.

Als het scherm binnen die 65280 bytes past (64×64 en 128×128 wel, maar 256×256 niet), begint het scherm bij adres 0, rij voor rij. Wat overblijft is het spriteblad dat is geselecteerd door `0x5F41` (0 is blad 0). Als dat blad groter is dan de resterende ruimte, selecteert `0x5F42` de pagina en wordt de kaart niet in het venster weergegeven. Als het blad wel past, bestaat het resterende deel uit kaartcellen van de kaart die zijn geselecteerd door `0x5F43`, één of twee bytes per stuk, en `0x5F44` verplaatst de kaart naar een andere pagina als deze niet in het resterende deel past.

Als het scherm niet past, is het hele venster één pagina van 65280 pixels. `poke(0x5F40, n)` draait de schermpagina om. `poke(0x5F41, n)` met een andere waarde dan 0 toont blad n in plaats daarvan, en blad 0 wordt in deze modus niet weergegeven. `poke(0x5F43, n)` met een andere waarde dan 0 toont kaart n, en die heeft voorrang op blad 0. Kaart 0 is in deze modus alleen zichtbaar als `mget` en `mset`.

Een `peek` die niet op een pixel of register terechtkomt, retourneert 0. Een `poke` op die plek doet niets. Sprite-vlaggen, SFX en DATA-tabellen worden niet in dit venster weergegeven. Gebruik `fget`, `sfxget` en `dget`.

Praktisch gebruik

Elke functie staat onderaan deze pagina vermeld, samen met praktische toepassingen van de PIX37-functies. De links onder elk voorbeeld geven meer informatie over de betreffende functies.



Vloeiende beweging

Een spelcartridge van 128×128 pixels, van bovenaf gezien. Sprite 1 is al getekend. Pijlen versnellen in elke richting en wrijving vertraagt je. De sprite loopt door: een helft die aan de onderkant loslaat, wordt aan de bovenkant getekend, en hetzelfde geldt voor de andere randen.

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



Controleren of de volgende zet geblokkeerd is

Een spelcartridge van 64x64 pixels. De vloer en de rechtermuur staan al op de kaart, ingekleurd met sprite 3, en vlag 0 is op die sprite ingesteld. Pijlen bewegen sprite 1. Het woord bovenaan geeft 'geblokkeerd' aan wanneer de volgende stap een effen tegel zou overlappen, en 'vrij' wanneer dat niet het geval is.

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



Een doekje met een draai

Druk op A. De wipe wordt getekend door flip(), één zwarte rij per keer, in plaats van binnen _draw(). _draw blijft op de achtergrond terwijl dat gebeurt, zodat elke rij op het scherm blijft.

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



Een tabel met functies (klasse)

Sprite 1 is al getekend. make_monster werkt als een klasse. Het retourneert een tabel, en update en draw zijn methoden die de x- en y-coördinaten van dat monster bijhouden. Het getal bovenaan geeft aan hoe ver het heeft gelopen.

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



Vier manieren om een knop te lezen

A is knop 4. Houd hem ingedrukt en de batterij loopt op. Laat hem los en de batterij gaat terug naar 0. Omhoog en omlaag drukken herhaalt zich terwijl je de knop ingedrukt houdt, zodat de menubalk blijft bewegen. btnd is het eerste frame van een druk, dat deze gamecartridge alleen gebruikt als het begin van die ingedrukte positie.

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



Schiet met de muis

Sprite 1 is al in het midden getekend. Een witte stip is de cursor. Met een linksklik wordt een pixel ernaartoe gestuurd. Het wieltje verandert de snelheid, die in de hoek staat afgedrukt.

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



Voer een naam in en bevestig deze.

Druk op A om het veld te openen. Omhoog en omlaag veranderen een knipperende letter. A schrijft de letter, X verwijdert hem, Y is een spatie en B bevestigt. Vervolgens verschijnt je naam op de regel.

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



Camera op de speler

Een spelcartridge van 128×128 pixels. Sprite 1 begint in het midden van een vloer van tegels. Pijlen bewegen. De camera houdt de sprite gecentreerd. De x- en y-coördinaten in de hoek scrollen niet, omdat camera() zonder argumenten het beeld terugzet voor de HUD.

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



Een paar regels uit een visuele roman

De tekstregels staan in DATA, in het Engels en Pools. A betekent 'doorgaan' of 'blijven' wanneer er een keuze op het scherm staat. B betekent 'vertrekken' bij een keuze en wisselt anders van taal. X wisselt van taal terwijl er een keuze wordt weergegeven. De dialoog gebruikt een lettertype van 8px, waardoor Poolse letters correct worden weergegeven.

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



Schilderen met de peek-and-prik-techniek.

Een gamecartridge van 64x64 pixels, dus adres 0 is de pixel linksboven en de volgende rij begint bij 64. De gamecartridge schrijft pixels, kopieert een rij, wist een rij en print de gelezen waarden terug.

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



Een lijst waaraan je dingen kunt toevoegen

A telt een getal op, tot en met 8. B verwijdert het laatste getal. X verwijdert het eerste getal. De lijst wordt van boven naar beneden op het scherm weergegeven.

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



Een naam ontleed

De spelcartridge splitst ada,12,blue en print de naam, de eerste letter ervan, 12, en het woord blue.

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



Speel een noot en verander die vervolgens.

Patroon 0 bestaat uit één noot, en elke herhaling heeft hetzelfde volume. A speelt het patroon opnieuw af zodra de noot is afgelopen. B verhoogt de toonhoogte en speelt die noot.

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



Een liedje dat in de gamecartridge is geschreven.

Twee patronen herhalen zich op rij 0, één op elk van de eerste twee kanalen. Rij 0 blijft 0. ch0 en ch1 zijn de noten waarop elk kanaal is ingesteld. Door het patroon, LEN en LOOP in de gamecartridge te bewerken, kun je zien wat deze instellingen doen in het muziekpaneel.

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



Drie monsters in DATA

Het DATA-paneel heeft al een kolom voor de naam en een kolom voor de levenspunten (hp), met slijm, vleermuis en oog. Een A doet het slijm pijn. De getallen aan de rechterkant geven de levenspunten (hp) aan.

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



Onthoud de score.

A telt een punt op en slaat het op. Sluit het spel af en open de cartridge opnieuw: het getal staat er nog steeds. B vergeet het.

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




Een tweede hoofdstuk in een ander bestand

Twee gamecartridges delen de savegame-functie. Deel 1: A geeft aan dat het spel is uitgespeeld. Deel 2, dat als eerste wordt geopend, geeft aan dat deel 1 moet worden uitgespeeld. Daarna opent deel 2 en wist B de savegame, zodat deel 1 opnieuw kan worden gespeeld.

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



Eén prestatie

Sprite 8 is al getekend. Druk tien keer op A. De melding geeft de prestatie weer en sprite 8 blijft staan omdat deze is ontgrendeld. Druk op B om die prestatie in het lokale bestand te wissen en de teller te resetten.

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



Een kamer waar je in kunt lopen.

Een spelcartridge van 256×256 pixels. Een witte stip is de cursor. Het menu bestaat uit 'host', 'deelnemen' of de serverlijst. 'Host' opent een kamer genaamd 'walk' en start het spel. 'Deelnemen' scrollt een adres (cijfers, een punt, een dubbele punt) en 'B' verbindt. De serverlijst beweegt met de pijltjes omhoog en omlaag, en 'A' voegt zich bij de rij. In het spel bewegen pijltjes sprite 1. Er wordt één munt gedeeld: de host verplaatst deze, en het oppakken ervan telt 1 op bij de score van die speler en zorgt voor de volgende sprite. Elke sprite, inclusief die van jou, toont de score ernaast. In de hoek wordt weergegeven hoeveel spelers er zijn, het LAN-adres en het openbare adres van de host. 'B' vertrekt: jij en de anderen keren terug naar het menu en de volgende host start een nieuwe kamer met score 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`



Een dialoog die wacht.

A toont de volgende regel. De functie pauzeert bij elke 'yield' totdat je weer op A drukt.

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



Wacht een halve seconde

Het woord verandert na ongeveer een halve seconde van ... naar hi. Het scherm blijft tekenen terwijl de coroutine gepauzeerd is.

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



Cirkels die één voor één aankomen

Vijf cirkels druppelen van bovenaf naar beneden, met telkens twee frames ertussen, en vallen dan weer naar beneden.

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



Twee frames, van de klok

Sprite 1 en sprite 2 zijn twee frames van hetzelfde personage. De afbeelding wisselt vier keer per seconde.

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

Elke functie

Argumenten tussen vierkante haken zijn optioneel. Indexen van tabellen en tekenreeksen beginnen bij 1, behalve DATA-rijen en -kolommen, die bij 0 beginnen. Kleuren zijn paletindexen.

Index

_teken
_init
_mp
_mp_fail
_mp_join
_mp_leave
_mp_update
_set_fps
_bijwerken
buikspieren
achhelder
Achdef
bereiken
bereikt
toevoegen
allemaal
beweren
atan2
bandje
bnot
bor
bn
btn
btnp
btnu
bxor
camera
plafond
Chr
omg
circfill
klikl
klikm
klikr
clip
cls
co-creëren
kleur
hoofdsom
want
costatus
tellen
cursor
dcols
del
delicatessenwinkel
delsave
dget
slaapt
dset
dstr
fget
vulp
omdraaien
fl
voorleggen
fset
informatie
iparen
sleutel uit
toetsaan
sleutelstat
toetsstr
lijn
laden
loggen
lshr
kaart
maximaal
memcpy
memset
mget
midden
min
muis
mset
muziek
muziekstat
aankondigen
bel
volgende
nhost
doe mee
nkb
nkbstr
nl
nllaat
nllijst
nlluister
opmerking
nspelers
nzend
nstun
ord
ovaal
ovale vulling
pak
paren
vriend
palrgb
palt
pcal
kijk
kijkje2
kijkje4
perfect
pagina
porren
porren2
porren4
afdrukken

afdrukken10	setpalet	liedset
afdrukken8	setpr	gesplitst
pset	settex	spr
rect	zet	sqrt
rectfill	sfx	strand
rnd	sfxget	sset
rot	sfxhdr	sspr
rot	sfxset	stat
gelijk	sfxstat	sub
correct invullen	sget	tijd
opslaan	sn	lijn
opslagslot	shl	tonum
scherm	showstatistieken	tostr
scroll	shr	uitpakken
selecteren	zonde	whoami
setmap	liedje krijgen	opbrengst

Levenscyclus

_init

`_init() ... end`

Wordt eenmaal uitgevoerd wanneer de gamecartridge start. Code die buiten een hook staat, wordt ook opgenomen in _init.

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

_update

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

Spellogica. Wordt aangeroepen met een frequentie van 60 Hz, zelfs wanneer _draw is gelimiteerd of de host trager is. Lees hier de knoppen.

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

_draw

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

Tekent één frame. Zonder argument, of met 0, wordt het zo vaak uitgevoerd als de host aangeeft. Geef n (1..999) door om het tekenen te beperken tot n frames per seconde. _update blijft op 60 Hz.

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

_set_fps

```
_set_fps(n)
```

Wijzigt de _draw cap tijdens het spelen. 0 betekent onbeperkt. n wordt begrensd tot 0..999.

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

flip

```
flip()
```

Toont het scherm en wacht tot het volgende frame. Gebruik dit alleen als je je eigen frame-loop tekent. Wacht er niet actief op na gebruik.

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

time

```
time()
```

Seconden sinds _init.

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

Invoer

btn

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

1 zolang knop b ingedrukt is, anders 0. p is de speler, 0..7. Zonder argumenten retourneert het een bitmasker van speler 0. Knoppen: 0 links, 1 rechts, 2 omhoog, 3 omlaag, 4 A, 5 B, 6 X, 7 Y, 8 R, 9 L, 10 selecteren, 11 start.

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

btnd

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

Dit geldt alleen voor het _update-frame wanneer knop b wordt ingedrukt.

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

btnp

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

Bij indrukken werkt het, en vervolgens nogmaals zolang de knop ingedrukt wordt gehouden: 30 ticks vertraging, daarna elke 8 ticks. Zonder knop wordt een masker weergegeven.

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

btnu

`btnu(b, [p])`

Dit geldt alleen voor het frame waarin de knop wordt losgelaten.

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

mouse

`mouse()`

Geeft de x- en y-coördinaten van de muis weer in spelpixels.

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

clickl

`clickl([p])`

Linkermuisknop. clickl() wordt ingedrukt gehouden. clickl(1) is de drukrand.

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

clickr

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

Rechtermuisknop. Dezelfde 1-argument rand als clickl.

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

clickm

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

De middelste knop, of een klik met het scrollwiel.

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

scroll

```
scroll()
```

Muiswielbeweging in dit frame. Positief en negatief zijn afhankelijk van de draairichting van het wieltje.

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

keyon

```
keyon()
```

Opent een tekstveld dat wordt bediend met knoppen. Omhoog en omlaag veranderen een letter van A tot en met Z. A typt de letter. X verwijdert. Y is een spatie. B bevestigt en sluit het veld. Het toetsenbord typt er niet in.

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

keyoff

`keyoff()`

Verbergt het tekstveld.

```
keyoff()
```

keystr

`keystr([s])`

Typ eerst tekst in het veld, dan bevestigt 1 op frame B, waarna de letter omhoog en omlaag verandert. Die letter is leeg zodra het veld sluit. keystr("") wist de tekst.

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

keystat

`keystat()`

1 zolang het tekstveld open is, anders 0.

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

Tekening

screen

`screen()`

Huidige schermgrootte van de gamecartridge in pixels. Geeft eerst de breedte en vervolgens de hoogte weer. Dit is de resolutie van het spel, niet van het editorvenster.

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

cls

`cls([col])`

Maakt het scherm leeg. De standaardkleur is 0.

```
cls(1)
```

color

`color(col)`

Stelt de tekenkleur in die wordt gebruikt wanneer een latere aanroep de parameter `col` weglaat. Zonder argument retourneert de functie de huidige kleur.

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

camera

`camera([x, y])`

Verschuift bij elke tekenoproep. camera() wordt gereset naar 0, 0.

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

clip

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

Beperkt het tekenen tot een rechthoek. Een niet-nul vijfde argument snijdt de huidige clip. clip() herstelt het volledige scherm. Het bekijken van de clipregisters is alleen-lezen.

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

cursor

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

Hiermee wordt de volgende afdruk zonder coördinaten ingesteld, en optioneel de kleur. cursor() retourneert x, y.

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

pset

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

Stelt één schermpixel in. Telt mee voor de pixelwaarde.

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

pget

`pget(x, y)`

Kleur van de schermpixel op positie x, y.

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

print

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

Tekent tekst van 3x5 inch (UTF-8, bijgesneden uit het Fusion-lettertype). Zonder x- en y-coördinaten wordt de tekst op de cursorpositie afgedrukt en gaat de cursor naar de volgende regel.

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

log

`log(...)`

Voegt de argumenten, gescheiden door spaties, toe aan een tekstbuffer. pix37_run print die buffer naar de standaarduitvoer. Het tekent niet.

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

print8

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

Hetzelfde als de printversie, met het 8 px Fusion-lettertype.

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

print10

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

Hetzelfde als de printversie, maar dan met het Fusion-lettertype van 10 px.

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

line

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

Lijn van x0,y0 naar x1,y1. Twee argumenten lopen door vanaf het vorige einde. line() reset die keten.

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

circ

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

Omtrek van een cirkel. De straal is in pixels.

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

circfill

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

Gevulde cirkel.

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

oval

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

De omtrek van de ellips bevindt zich binnen het kader. De hoeken zijn inbegrepen.

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

ovalfill

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

Gevulde ellips binnen een kader.

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

rect

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

Rechthoekige omtrek. De hoeken zijn inbegrepen.

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

rectfill

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

Gevulde rechthoek.

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

rrect

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

Omtrek van een afgeronde rechthoek. x, y is de linkerbovenhoek. w en h zijn de afmetingen in pixels; beide moeten groter zijn dan 0. r is de hoekradius en is standaard 0. Deze wordt begrensd tot de helft van de kortste zijde.

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

rrectfill

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

Gevulde afgeronde rechthoek. Dezelfde argumenten als rrect.

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

fillp

`fillp([p])`

16-bits vulpatroon voor later gevulde vormen. 0 is effen. fillp() retourneert het huidige patroon. Bits doorlopen het 4×4-patroon.

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

pal

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

Wijst kleur c0 opnieuw toe aan c1. Modus 0 is het tekenpalet, 1 het weergavepalet, 2 het vulpatroonpalet. pal() reset.

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

palrgb

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

Stelt één paletitem in op RGB, elk kanaal 0..255.

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

palt

`palt([col, t])`

Transparantie. palt(col) of palt(col, true) maakt col transparant. palt(col, false) maakt het dekkend. palt() maakt elke kleur dekkend.

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

setpalette

`setpalette(n)`

Selecteert paletbank n. setpalette() retourneert de huidige bank.

```
setpalette(1)
```

tline

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

Tekent een lijn op het scherm terwijl kaartpixels worden bemonsterd vanaf kaartpositie mx, my, mdx, mdy. Verplaatst de kaartbemonstering langs de lijn.

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

Sprites en kaarten

spr

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

Tekent sprite n op x, y. w en h zijn de afmetingen in pixels, standaard 8. fx en fy spiegelen horizontaal en verticaal wanneer ze niet nul zijn. tex is het tabblad van het sprite-sheet. rot is de hoek in omwentelingen (0..1), dezelfde richting als sin en cos. 0 laat de sprite ongeroteerd.

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

sspr

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

Kopieert een rechthoek van het spriteblad naar het scherm, geschaald naar dw bij dh. Standaard behouden ze de oorspronkelijke afmetingen.

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

sget

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

De kleur van één texel op het spriteblad.

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

sset

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

Schrijft één texel op het spriteblad.

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

fget

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

Sprite-vlaggen. fget(n) retourneert de vlagbyte. fget(n, f) is waar als vlagbit f (0..7) is ingesteld.

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

fset

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

Stelt sprite-vlaggen in. fset(n, v) schrijft de hele byte. fset(n, f, v) stelt één bit in.

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

setspr

```
setspr(n)
```

Selecteert het actieve spriteblad. setspr() retourneert het huidige blad. Dezelfde bank als settex.

```
setspr(1)
```

settex

```
settex(n)
```

Alias van setspr.

```
settex(0)
```

map

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

Tekent de kaart. Cel cx, cy van de kaart wordt getekend op scherm sx, sy. cw, ch van 0 betekent de hele kaart. Layer wordt geaccepteerd en genegeerd. tex is het tegelsetblad.

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

mget

`mget(cx, cy)`

Sprite-ID in kaartcel cx, cy.

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

mset

`mset(cx, cy, n)`

Schrijft sprite n naar kaartcel cx, cy.

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

setmap

`setmap(n)`

Selecteert kaart n. setmap() retourneert de huidige kaart. Kaart 0 is een echte kaart.

```
setmap(1)
```

Geheugen

peek

```
peek(addr)
```

Leest één byte van het 64 KB-venster. Het venster is het actieve scherm, vervolgens de spritepixels en daarna de kaartcellen wanneer het scherm in 65280 bytes past. Grotere schermen worden gepagineerd met poke(0x5F40, page). Adressen die niet worden gemapt, retourneren 0. Ook geschreven naar @addr.

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

poke

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

Schrijft bytes naar adres addr. Extra argumenten schrijven de volgende bytes. Niet-toegewezen adressen worden genegeerd. Ook $addr en val worden geschreven. Het aanraken van het scherm telt als een pixelschrijfbewerking.

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

peek2

```
peek2(addr)
```

Niet-ondertekende 16-bits waarde, little-endian, op adres.

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

poke2

```
poke2(addr, val)
```

Schrijft een niet-ondertekende 16-bits waarde, little-endian.

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

peek4

```
peek4(addr)
```

Niet-ondertekende 32-bits waarde, little-endian.

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

poke4

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

Schrijft een niet-ondertekende 32-bits waarde, little-endian.

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

memcpy

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

Kopieert len bytes binnen hetzelfde venster van 64 KB.

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

memset

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

Vult len bytes vanaf dest met val.

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

Wiskunde

abs

`abs(x)`

Absolute waarde.

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

flr

`flr(x)`

Afrondingen richting min oneindigheid.

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

ceil

`ceil(x)`

Afrondingen richting plus oneindigheid.

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

max

`max(a, b)`

De grootste van de twee waarden.

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

min

`min(a, b)`

De kleinste van de twee waarden.

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

mid

`mid(a, b, c)`

De middelste van de drie waarden. Handig als klem.

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

sgn

`sgn(x)`

Teken van x: -1, 0 of 1.

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

sqrt

`sqrt(x)`

Wortel. Een negatief argument is geen getal.

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

sin

`sin(x)`

Sinus. x wordt in stappen weergegeven: 0, 0,25, 0,5, 0,75, één keer rond de cirkel. Niet in radialen.

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

cos

`cos(x)`

Cosinus, op zijn beurt, dezelfde conventie als sinus.

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

atan2

`atan2(dx, dy)`

De hoek van de vector dx, dy, verandert van 0 tot 1.

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

rnd

`rnd([x])`

Willekeurig getal in [0, x). Standaard x is 1, dus rnd() is 0..1. Voor een dobbelsteen: flr(rnd(6))+1.

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

srand

`srand(seed)`

De randomgenerator wordt geïnitialiseerd zodat een latere rnd-reeks herhaald kan worden.

```
srand(1)
```

band

`band(a, b)`

Bitwise AND. Ook wel geschreven als a & b.

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

bor

`bor(a, b)`

Bitwise OR. Ook wel geschreven als a | b.

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

bxor

`bxor(a, b)`

Bitwise XOR. Ook geschreven als a ^^ b.

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

bnot

`bnot(a)`

Bitwise NOT. Ook geschreven als ~a.

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

shl

`shl(a, n)`

Verschuif naar links. Ook geschreven als << n.

shr

`shr(a, n)`

Rekenkundige verschuiving naar rechts. Ook geschreven als >> n.

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

lshr

`lshr(a, n)`

Logische verschuiving naar rechts. Ook geschreven als >>> n.

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

rotl

```
rotl(a, n)
```

Draai de bits naar links. Ook geschreven als <<> n.

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

rotr

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

Draai de bits naar rechts. Ook geschreven als >>< n.

Tabellen en waarden

add

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

Voegt v toe aan tabel t. Als i is opgegeven, wordt ingevoegd op die index (beginnend bij 1).

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

del

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

Verwijdert de eerste waarde gelijk aan v.

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

deli

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

Verwijdert index i en retourneert die waarde. Zonder i wordt de laatste invoer verwijderd.

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

all

```
all(t)
```

Iterator over het arraygedeelte, in volgorde. Gaten stoppen de iteratie, net zoals bij een array die begint met 1.

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

foreach

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

Roept fn(v) aan voor elk element van de array.

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

pairs

```
pairs(t)
```

Iterator over elke sleutel en waarde, inclusief benoemde velden.

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

ipairs

```
ipairs(t)
```

Iterator over het arraygedeelte met de 1-gebaseerde index.

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

next

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

Geeft de volgende sleutel en waarde na k terug. next(t) begint bij het begin. Wordt gebruikt door paren.

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

count

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

Lengte van het array-gedeelte. Als v gegeven is, telt het hoeveel elementen gelijk zijn aan v.

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

pack

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

Verpakt de argumenten in een nieuwe tabel.

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

unpack

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

Retourneert t[i] tot en met t[j] als afzonderlijke waarden. Standaardwaarden hebben betrekking op het hele arraygedeelte.

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

select

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

select("#", ...) geeft aan hoeveel argumenten er volgen. select(i, ...) geeft de argumenten vanaf i weer.

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

assert

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

Stopt met een foutmelding wanneer c onwaar of nil is. msg vervangt de standaardtekst.

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

pcall

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

Roept f aan. Bij succes retourneert de functie true en de resultaten. Bij een fout retourneert de functie false en de foutmelding.

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

Snaren

chr

```
chr(n)
```

Eén teken uit een bytecode van 0..255.

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

ord

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

Bytecodes van s van index i tot j. Indexen zijn 1-gebaseerd. Standaard wordt het eerste teken gebruikt.

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

sub

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

Deeltekenreeks van i tot en met j, beginnend bij 1. Laat j weg om het laatste deel te nemen.

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

split

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

Splitst s in een tabel. sep is standaard een komma. Een getal sep is de breedte van een chunk. conv converteert stukken met tonum wanneer dit waar is.

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

tonum

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

Parseert een getal. Het grondgetal kan 2 tot en met 36 zijn. Retourneert nil als de tekst geen getal is.

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

tostr

`tostr(v, [digits])`

Zet een waarde om in tekst. digits stelt het aantal decimalen voor getallen in.

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

Audio

sfx

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

Speelt patroon n af en retourneert het kanaal. Laat ch weg, of geef -1 door, en het eerste vrije kanaal wordt gekozen. ch -2 stopt patroon n op elk kanaal. n -1 stopt een kanaal, of elk kanaal wanneer ch wordt weggelaten. n -2 laat een herhalend patroon eindigen in plaats van te herhalen. sfx() zonder argumenten retourneert -1.

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

sfxget

`sfxget(n, step)`

Geeft een tabel terug met één patroonstap: [1] toonhoogte, [2] golf, [3] volume, [4] effect. Toonhoogte is 0..63 (C-0 tot en met D#5). Golf is 0..15. Volume is 0..7, en 0 is stil. Effect is 0..7.

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

sfxset

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

Schrijft één patroonstap. Alle zes argumenten zijn verplicht.

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

sfxhdr

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

Patroonheader. sfxhdr(n) retourneert de headertabel. sfxhdr(n, key, val) stelt één veld in: snelheid, loop_in, loop_out, niveau, ruis, crush, dtune, rverb, dampn, swing, stappen. De getallen 0..10 selecteren dezelfde velden in die volgorde. Snelheid is TIC, ticks per stap. Hoger is langzamer. 15 is 120 beats per minuut. 16-bit accepteert 1..255. Nauwere synthesizers springen naar hun TIC-lijst.

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

sfxstat

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

Patroon-ID wordt afgespeeld op kanaal ch, of -1. Standaardkanaal is 0.

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

note

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

Speelt één noot zonder patroon. Toonhoogte 0..63, golfvorm 0..15, volume 0..7, effect 0..7. 'ticks' geeft aan hoe lang de noot aanhoudt, in dezelfde 120 Hz-tikken als TIC. Laat 'ticks' weg en de noot duurt 15 seconden, één standaardstap. Het effect loopt gedurende die tijd, waarna het kanaal stilvalt.

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

music

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

Start nummer n vanaf het begin. music(-1) stopt. Als je music aanroept met de id van een nummer dat al speelt, blijft het nummer doorspelen. music() retourneert de id van het spelende nummer, of -1. fade wordt weergegeven in milliseconden: het nummer begint of eindigt wanneer n -1 is. De ingestelde maskbits geven aan welke kanalen voor het nummer worden gebruikt. 0 betekent alle acht kanalen. De overige kanalen blijven vrij voor geluidseffecten.

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

musicstat

```
musicstat()
```

Tabel van het afspelende nummer. 'song' en 'row' zijn de nummer-ID en de rij, of -1 als er niets wordt afgespeeld. 'step' en 'pos' bestaan uit acht getallen. 'step[1]' is kanaal 0, 'step[8]' is kanaal 7. 'step' is de nootindex, of -1 als dat kanaal stil is. 'pos' geeft aan hoe ver die noot is gegaan, van 0 tot 1. 'tick' geeft aan hoeveel ticks deze rij al heeft afgespeeld. 'played' geeft aan hoeveel rijen er zijn gestart sinds 'music()', inclusief de rij die momenteel wordt afgespeeld.

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

songget

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

Patroon-ID opgeslagen in een songcel. ch is 0..7. 255 betekent dat de cel leeg is. Dit is de live-aanroep; de codehint toont nog steeds een oudere argumentvolgorde.

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

songset

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

Schrijft een patroon-ID in een songcel. Gebruik 255 om de cel te wissen.

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

stat

```
stat(n)
```

Alleen audio. 16..23 is de sfx-ID op kanaal n-16. 24..31 is het muziekpatroon op kanaal n-24. Elke andere n retourneert 0. Frametiming en RAM zijn prestatie- en informatiegegevens, geen statistieken.

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

Gegevenstabellen

dget

`dget(t, row, col)`

Leest een cel uit de gegevenstabel t in het paneel DATA. Rij en kolom beginnen bij 0.

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

dset

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

Schrijft een cel. Gebruik een tekenreeks voor een tekstkolom.

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

dstr

`dstr(t, row, col)`

De cel als tekst. Getallen worden geconverteerd.

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

drows

`drows(t)`

Hoeveel rijen heeft tabel t?

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

dcols

`dcols(t)`

Hoeveel kolommen heeft tabel t?

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

Opslaan en prestaties

save

`save(k, v)`

Slaat v op onder sleutel k in het opslagbestand. v kan een getal, een tekenreeks of een tabel daarvan zijn. Het bestand is gecomprimeerd.

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

load

`load(k)`

Leest sleutel k. Retourneert nil als de sleutel ontbreekt.

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

delsave

`delsave(k)`

Verwijdert de toets 'k' uit het opslagbestand.

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

saveslot

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

Selecteert het opslagbestand in de map van de auteur. Als er geen naam is, wordt de speltitel gebruikt. Retourneert 1 als dat bestand al bestaat.

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

achieve

```
achieve(i)
```

Ontgrendelt prestatie i, van 0 tot 15. Lokaal opgeslagen. Een Steam-build stelt ook ACH_i in. Een catalogussessie rapporteert dit aan het spelersaccount.

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

achieved

```
achieved(i)
```

1 wanneer prestatie i al is ontgrendeld.

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

achdef

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

Toasttekst voor prestatie i, die 15 seconden lang onderaan het scherm wordt weergegeven. spr is een optionele 8×8 tegel, tex is het bijbehorende vlak.

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

achclear

```
achclear(i)
```

Verwijdert prestatie i in het lokale bestand. Een Steam ACH_i die al was ingesteld, blijft ingesteld.

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

Netwerk

whoami

```
whoami()
```

Je spelers-ID is 0..15 zolang je sessie actief is. Offline is deze -1.

```
id = whoami()
```

nhost

```
nhost()
```

1 als deze machine de host is, anders 0.

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

nplayers

```
nplayers()
```

Hoeveel live-ID's bestaan er in het bereik 0..15?

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

nlisten

```
nlisten()
```

Start met het hosten van UDP en retourneert het LAN-adres als ip:poort.

```
addr = nlisten()
```

njoin

```
njoin(addr)
```

Maakt verbinding via IP-adres of IP-adres:poort.

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

nleave

```
nleave()
```

Verlaat de sessie. Peers zien _mp_leave met reden 0.

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

nsend

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

Verstuurt een bericht met verlies van gegevens. De naam is de gebeurtenis. Maximaal vier extra waarden volgen. p van 0 tot 15 fluistert naar die peer; laat p weg om naar iedereen te verzenden.

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

ncall

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

Dezelfde structuur als nsend, maar het bericht wordt opnieuw verzonden totdat het bevestigd is. Drie mislukte pogingen achter elkaar beëindigen de sessie.

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

nlan

`nlan()`

Huidig LAN-IP-adres:poort na nlisten.

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

nstun

`nstun()`

Uw openbare IP-adres:poort van STUN. Een gast op een ander netwerk kan proberen verbinding te maken.

```
pub = nstun()
```

setturn

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

TURN-terugvaloptie wanneer een direct pad mislukt. setturn() met lege argumenten schakelt deze functie uit. Het adres is host:poort.

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

nlist

`nlist(game)`

Tabel met beschikbare sessies voor die spelnaam. Elk item bevat het adres, het aantal spelers en de naam van het spel.

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

nannounce

`nannounce(game)`

Hartslag, zodat deze host in nlist verschijnt voor het spel. Roep dit aan met een timer tijdens het hosten.

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

nkb

`nkb([on])`

Adresveld. nkb(1) opent het, nkb(0) verbergt het, nkb() geeft aan of het open is. Omhoog en omlaag veranderen een teken, 0-9 of een punt of een dubbele punt. A typt het. X verwijdert. Het toetsenbord typt er niet in.

```
nkb(1)
```

nkbstr

`nkbstr([s])`

Tekst in het adresveld, waarna de tekens omhoog en omlaag veranderen. nkbstr("") wist de tekst.

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

_mp

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

Je schrijft deze functie. Deze wordt uitgevoerd wanneer een bericht binnenkomt. 'from' is de afzender-ID. 'name' is de gebeurtenis. Er kunnen maximaal vier waarden volgen.

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

_mp_join

`_mp_join(p)`

Jij schrijft deze functie. Een peer ontving id p. Deze functie wordt ook voor jou geactiveerd; whoami() is die id.

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

_mp_leave

`_mp_leave(p, why)`

Je schrijft deze functie. Peer p is vertrokken. Waarom is 0 een normaal vertrek? Waarom is 1 een verbroken verbinding?

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

_mp_update

`_mp_update()`

Je schrijft deze functie. Deze wordt elke 15 _update frames uitgevoerd, ongeveer 4 Hz. Een goede plek om posities te verzenden.

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

_mp_fail

`_mp_fail()`

Je schrijft deze functie. De sessie is verbroken: de host is verdwenen of ncall is drie keer mislukt.

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

Coroutines

cocreate

`cocreate(f)`

Maakt een coroutine van functie f. Deze wordt pas uitgevoerd na het hervatten van de coresume.

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

coresume

`coresume(c, ...)`

Voert coroutine c uit totdat deze een waarde retourneert. Bij succes: true en de geretourneerde waarden. Bij een fout: false en de fout.

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

costatus

```
costatus(c)
```

"opgeschort", "lopend" of "dood".

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

yield

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

Geeft de controle terug aan degene die coresume heeft aangeroepen. De waarden worden het resultaat van die coresume.

```
yield(1)
```

Meeteenheid

info

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

Tekent de live HUD in _draw: fps (gemiddelde over 1 seconde), upd, drw, px, max, ram, peak, ramc en net wanneer een sessie actief is. info() gebruikt de printcursor. info(col) stelt alleen de kleur in. info(x, y) of info(x, y, col) plaatst het object. Getallen gebruiken B, k of m zodat ze op een klein scherm passen. De aanroep zelf wordt niet meegeteld in px.

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

showstats

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

Dezelfde HUD als info, automatisch getekend op 0, 0 na _draw, waarbij camera en clip worden genegeerd. showstats(0) verbergt hem. showstats(1) of showstats() toont hem.

```
showstats(1)
```

perf

`perf(n)`

Eén HUD-nummer, zodat je het zelf kunt tekenen. 0 fps, 1 upd, 2 drw, 3 px, 4 max (zwaarste frame op dit moment), 5 ram, 6 ramc (ram plus de ingepakte gamecartridge), 7 piek ram sinds de gamecartridge is gestart.

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