Manuál PIX37

Nejprve editor, poté kód. Offline kopie knihy na pix37.party/docs.

Editor

PIX37 DEV je editor. Zde kreslíte, píšete a posloucháte hru a poté exportujete herní cartridge nebo samostatnou sestavu. Umí také přehrát herní cartridge a otevřít PIX37.PARTY pro stahování herních cartridge a používání katalogu. PIX37 DEV EDUCATIONAL je stejný editor a je zdarma. Nemá žádné sestavení, nemá export herních cartridge a nemá přístup k PIX37.PARTY. PIX37 Console je přehrávač. Přehrává herní cartridge `.p37` a vy otevřete PIX37.PARTY pro stahování herních cartridge a procházení katalogu.

Projekty a herní cartridge

Soubor `.pix37` je projekt: zdrojové karty, sprajty, mapy, zvuk a datové tabulky. Soubor `.p37` je herní cartridge, kterou hráč spouští: hustě zabalený bajtkód.

Uložené soubory a achievementy se nacházejí ve složce autora vedle dat aplikace. Ve Windows to vypadá jako `C:\\Users\\Ada\\AppData\\Roaming\\PIX37\\Mira\\Pong.p37s` pro uložené soubory a `C:\\Users\\Ada\\AppData\\Roaming\\PIX37\\Mira\\Pong.ach` pro achievementy. V Linuxu jsou stejné soubory `~/.local/share/PIX37/Mira/Pong.p37s` a `~/.local/share/PIX37/Mira/Pong.ach` . Ada je účet v počítači. Mira je jméno autora a tento název je složka. Pong je slot pro ukládání, který je standardně nastaven na název hry. `save` zapíše soubor `.p37s` . První odemčený achievement zapíše soubor `.ach` . Pokud jsou soubory již na místě, běžec je přečte při spuštění herní kazety.

Vývojářské menu

Stisknutím klávesy Escape se otevře vývojářské menu. Stisknutím klávesy Escape během běžící hry se vrátíte do editoru. Potvrďte stisknutím klávesy Enter nebo tlačítka A (ve výchozím nastavení V). Stisknutím klávesy B (ve výchozím nastavení C) se vrátíte do editoru.

Pokud není načten žádný projekt, nabídka je následující: VYTVOŘIT PROJEKT, NAČÍST PROJEKT, SESTAVIT, NAČÍST Herní kazetu, PŘIPOJIT K PIX37.PARTY, NASTAVENÍ.

Po načtení projektu to bude: ULOŽIT PROJEKT, NAČÍST PROJEKT, VYTVOŘIT PROJEKT, AKTIVUM, SESTAVIT, NAČÍST Herní kazetu, PŘIPOJIT SE K PIX37.PARTY, NASTAVENÍ.

Ctrl+S uloží otevřený projekt. Ctrl+R jej spustí.

Vytvořte projekt

Příkaz CREATE PROJECT si před zahájením tvorby projektu vyžádá herní cartridge.

- **VIDEO** — 1-BIT (2 barvy), 2-BIT (4), 4-BIT (16), 8-BIT (256).
- **AUDIO** — hloubka samplování syntezátoru: 16, 8, 4, 2 nebo 1 bit.
- **OBRAZOVKA** – předvolby zahrnují 64×64, 128×128, 160×90, 256×256 a 320×180. VLASTNÍ velikost umožňuje vybrat libovolnou velikost od 1×1 do 512×512.
- **PALETY** — banky barev. První banka začíná níže uvedenou 16barevnou sadou. Příkaz NAČÍST PALETU použije vestavěnou sadu: PICO-8 16, PICO-8 32, Game Boy, 1-bit, CGA, Sweetie-16 a delší seznam převzatý z Lospecu (Resurrect 64, Apollo, Endesga 32 a 64, CC-29, SLSO8, Pear36, Oil 6, Duel, AAP-64, Lost Century a další). Můžete také načíst soubor palety nebo načíst vlastní PNG a paletu z daného obrázku vyjmout.



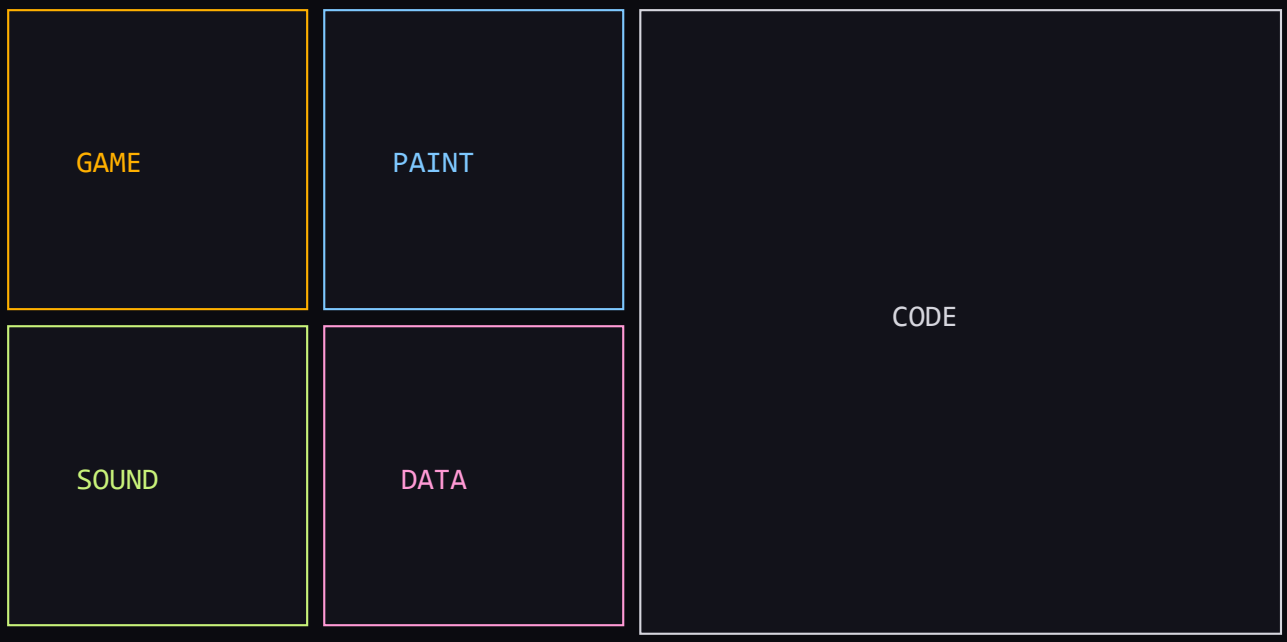
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

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

Barevné indexy používané funkcemi `cls` , `pset` , `spr` a ostatními. Jakýkoli index můžete překreslit v editoru palet.

Pět panelů

Editor je HRA, MALBA, KÓD, ZVUK a DATA, v tomto pořadí. Mapa se nachází v MALBĚ. SFEX a skladby se nacházejí v ZVUKU. Kliknutím na panel jej můžete otevřít. Když je fokus na KÓD, ZVUK a DATA ignorují klávesy s písmeny, takže psaní zároveň nehraje na klavír.



Plné rozvržení, psané `1:1:4:1:1` . Každé číslo udává, jak velkou část řádku daný panel zabírá: HRA, BARVA, KÓD, ZVUK, DATA. Malá obrazovka je umístěna uprostřed buňky o rozměrech 128×128. Obrazovka větší než 128 px rozšiřuje sloupec HRA.

NASTAVENÍ → GLOBÁLNÍ → ROZVRŽENÍ cyklicky mění uspořádání:

Kód	Co vidíš
`1:1:4:1:1`	Všechno. HRA a ZVUK naskládané na sebe, BARVA a DATA naskládané na sebe, KÓD vpravo.
`1:1:4:1:1 2X`	Stejné uspořádání. KÓD je vykreslen s dvojnásobnou hustotou, takže se vejde dvakrát tolik textu.
`1:0:1:0:1`	HRA, DATA, KÓD.
`1:0:1:0:1 2X`	Stejné uspořádání s hustším CODE.
`0:1:1:1:1`	Žádný herní náhled. MALBA, KÓD, ZVUK, DATA.
`0:1:1:1:1 2X`	Stejné uspořádání s hustším CODE.
`0:1:4:1:2`	Studio. Široký KÓD, vyšší DATOVÝ pás, BARVA a ZVUK.
`0:1:4:1:2 2X`	Stejné uspořádání s hustším CODE.

`0:0:4:0:4`	Pouze KÓD a DATA, obojí velké.
`0:0:4:0:4 2X`	Stejné uspořádání s hustším CODE.
`0:1:4:1:0`	BARVA, ZVUK a široký KÓD.
`0:1:4:1:0 2X`	Stejné uspořádání s hustším CODE.
`0:4:0:0:0`	Pouze MALBA, 256×256, jeden pixel listu na jeden pixel plátna.

Řádek `2X` zachovává stejné panely. Panel KÓD je vykreslen v dvojnásobném rozlišení a zmenšen zpět do stejné buňky, takže text, tabulátory a mezery mezi nimi jsou poloviční a na obrazovku se vejde dvakrát tolik zdrojového kódu.

HRA

Náhled spuštěné herní kazety. Klikněte na panel HRA nebo stiskněte Ctrl+R pro spuštění náhledu. Panel se konfiguruje v NASTAVENÍ → HERNÍ PANEL:

- **Náhled** – barva v nečinnosti, nepřetržitý náhled nebo automatické opětovné načtení při úpravách.
- **Hrajte v novém okně** – hra se otevře vedle editoru.
- **Zobrazení** – náhled jako RGB888 nebo RGB565.
- **HRA 64** — obrazovka o rozměrech 64×64 je buď vycentrována v poměru 1:1 v buňce, nebo je zvětšena 2× tak, aby ji zaplnila.
- **Shadery** — CRT a filtry ve hře, oddělené od filtrů v editoru.
- **Barva** — pozadí panelu.

MALOVAT

Listy spritu a na straně mapy ve stejném panelu dlaždicové mapy. Listy mohou být větší než obrazovka. Velikosti náhledu spritu jsou 8, 16, 32, 64 a 128 (128 pouze v případě, že list má alespoň 128 px). Otáčením kolečka myši nad náhledem spritu jej přiblížíte. Velikosti štětce jsou 1, 2, 3 a 4.

Klíč	Nástroj
B.	Tužka
M	Výběr obdélníku
Otázka	Laso
S	Razítko

G	Vědro
Já	Kapátko
PROTI	Pohyb
L / Shift+L	Čára / křivka
U / Shift+U	Dutý obdélník / dutá elipsa
F / Shift+F	Vyplněný obdélník / vyplněná elipsa
D	Váhat
[ ]	Menší / větší štětec
Ctrl+Z / Ctrl+Y	Zpět / Znovu. Ctrl+Shift+Z také opakuje.
Ctrl+C / Ctrl+X / Ctrl+V	Kopírovat, vyjmout, vložit
Ctrl+A	Vyberte plátno

Mapa využívá stejnou myšlenku, s menším počtem nástrojů: B tužka, M obdélníkový výběr, S razítko, V pohyb a Ctrl+Z, Ctrl+Y, Ctrl+C, Ctrl+V. Podržením klávesy Alt přiblížíte texturu mapy. Velikost štětce na mapě zůstává stejná jako u nástrojů pro kreslení.

Každý sprite má osm příznakových bitů. Funkce Paint je na sprite nastaví; `fget` a `fset` je ve hře načtou.

NASTAVENÍ → EDITOR KARBY je místo, kde se tento panel konfiguruje:

- **Paleta** — která banka je nakreslena, plus načtení a uložení.
- **Náhled** — RGB888 nebo RGB565.
- **Kurzor štětce** – ukazatel při malování.
- **Přichycení pixelů** – jak se štětec přichycuje k pixelům.
- **Velký štětec** – velký štětec se zacvakne z levého horního rohu nebo z celé své plochy.
- **Cibulová slupka** – duch jiných snímků a směr, kterým se tento duch dívá.
- **Přiblížení mapy** – skutečné pixely nebo bezplatné přiblížení.
- **Barvy** — barvy uživatelského rozhraní pro malování.

KÓD

Záložky jsou samostatné soubory v projektu. Kliknutím na název záložky ji přejmenujete; Enter potvrdí. Několik záložek, z nichž každá definuje `_init` , `_update` nebo `_draw` je spojeno dohromady v pořadí záložek: první záložka se spustí jako první, poslední záložka se spustí jako poslední. Kód, který není uvnitř hooku, se stává součástí `_init` . Herní cartridge bez hooků je považována za jeden `_draw` .

Klíč	Akce
Šipky	Přesunutí kurzoru. Shift rozšíří výběr.
Ctrl+Vlevo / Vpravo	Slovo.
Domů / Konec, Ctrl+Domů / Konec	Začátek nebo konec řádku, případně tabulátoru.
Záložka	Vložte dvě mezery. V seznamu pro dokončení přijměte název.
Zadejte	Nový řádek, odsazený jako předchozí. Přijímá doplnění.
Ctrl+C / X / V	Kopírovat, vyjmout, vložit.
Ctrl+Z / Ctrl+Y	Zpět / znovu.
Ctrl+A	Vyberte kartu.
Ctrl+D	Duplikujte čáru nebo výběr směrem dolů.
Ctrl+/	Komentovat nebo odkomentovat pomocí `--` .
Ctrl+F	Najít. F3 přeskočí na další zásah. Escape zavře lištu.
Ctrl+G	Přejít na číslo řádku.
Uniknout	Nejprve zavře hledání, přejděte na nebo seznam nápověd. Druhý stisk klávesy Escape opustí editor a vrátí se do vývojářské nabídky.

Při psaní názvu funkce se zobrazují nápovědy. Stisknutím klávesy Enter nebo Tab vložíte název a klávesou `(` pokud je v nastavení zapnuta možnost DOKONČIT). NASTAVENÍ → EDITOR KÓDU:

- **Hledat** – Ctrl+F na aktivní kartě nebo napříč všemi kartami.
- **Vysoký text** – oříznutí glyfů na 5 px nebo zobrazení celého glyfu.
- **Velikost řetězce** — velikost písma řetězcových literálů.
- **Písmo komentáře** – 3×5 nebo písmo Fusion aktuálního jazyka uživatelského rozhraní.
- **Dokončit (** — zda doplňování typuje závorku.
- **Nápovědy kódu** – zapnuto nebo vypnuto.
- **Jazyk nápověd** – jazyk nabídky nebo angličtina.
- **Mezera stříšky** — dodatečné posunutí na prázdných řádcích.
- **Okap** – čísla řádků.
- **Barva** — pozadí, komentáře, tabulace, čísla, operátory, klíčová slova, řetězce, funkce enginu, vaše funkce, proměnné, závorky, stříška, výběr, chybový řádek, varování.

ZVUK

Panel má dvě záložky. SFX je jeden pattern: krátká série not. SOUND je skladba: umisťuje tyto patterny na osm kanálů. TIC udává, kolik tiků jeden krok čeká. Takt je 120 tiků za sekundu a jeden krok je šestnáctinová nota, takže 15 tiků je 120 úderů za minutu. Vyšší TIC je pomalejší pattern. Dva patterny zůstávají v kroku, když jeden TIC dělí druhý: 15 a 30 ano, 15 a 8 ne.

Klavír hraje, když je nota vybrána myší. Bílé klávesy jsou ZXCVBNM (CDEFGAB). Černé klávesy jsou SDGHJ (C# D# F# G# A#). Najeďte myší na libovolnou hodnotu a zadejte číslo, nebo na ni najeďte myší a otáčejte kolečkem. Čísla výšek tónů v kódu se pohybují od 0 (C-0) do 63 (D#5).

V jednom patternu jsou LOOP a LEN stejná dvě čísla. OUT nad 0 se smyčkuje od IN do OUT a název zní LOOP. OUT v 0 a IN nad 0 je LEN: tento počet not se zahraje od začátku, poté pattern končí a název zní LEN. Oba v 0 se hrají každý krok.

Kolik syntetizátoru můžete upravovat, závisí na bitové hloubce zvolené pro herní kazetu. NASTAVENÍ → EDITOR SFX nastavuje tuto hloubku, zda se nota automaticky posouvá vpřed (vypnuto, zřetězení not nebo zřetězení polí), zda Space restartuje přehrávání nebo přepíná mezi spuštěním a zastavením a barvy editoru.

16bitový

Každý krok má notu, hlasitost, vlnu a efekt. Hlasitost je 0 až 7. 0 je ticho.

Vlny:

- **0 trojúhelník**
- **1 nakloněná pila**
- **2 pily**
- **3 čtvereční**
- **4 pulzy** – úzký pulz
- **5 varhan**
- **6 hluk**
- **7fázovač**
- **8 PWM** — čtverec, jehož šířka se pohybuje
- **9 pulzů 25 %**
- **10 sinus**
- **11 kovový zvuk**
- **Zvon 12 FM**
- **13 trhat**
- **14 superpil**
- **15 PWM (C64)**

Efekty na jednu notu:

- **0 žádný**
- **1 snímek** — výška tónu a hlasitost se v tomto kroku posunou k další notě

- **2 vibrato** — kolísání výšky tónu
- **3. pokles** — výška tónu klesá do ticha, obvyklý kop
- **4 zeslabování**
- **5 zeslabování**
- **6. arpeggio** — prochází tuto notu a další tři, jedno každých 8 tiků. Na notě 8 nebo nižší, každé 4 tiky.
- **7 rychlé arpeggio** — stejné čtyři noty, každé 4 tiky, nebo každé 2 tiky, když je TIC 8 nebo méně.

Záhlaví vzoru, nad poznámkami:

- **TIC** — počet tiků na krok, od 1 do 255. 15 je 120 tepů za minutu. Zadejte číslo nebo jej zvyšte o jeden krok.
- **LOOP IN / OUT** — s OUT nad 0 se pattern smyčkuje od IN do OUT. Ponechte OUT na 0 a název se změní na LEN: IN udává počet přehraných not a 0 znamená celý pattern.
- **ŠUM** — celý vzorec se stává šumem.
- **CRUSH** — bzučivá varianta každé vlny.
- **DTUNE** — 0 vypnuto, 1 lehký refrén, 2 silný refrén.
- **RVERB** — dozvuk: 0 vypnutý, 1 krátký, 2 dlouhý.
- **DAMPN** — dolnopropustný filtr: 0 vypnuto, 1 slabý, 2 silný.
- **LEVEL** — hlasitost celého patternu, 0 až 7.
- **SWING** – náhodné posouvání. Liché kroky čekají o něco déle, sudé kroky o něco kratší dobu.
- **KROKY** — počet kroků, které vzor používá: 32, 24, 16 nebo 12.

8bitový

Krok má notu a vlnu. Čtyři vlny jsou čtvercová, trojúhelníková, pilovitá a šumová. Záhlaví má TIC, LOOP IN / OUT, FX a LEVEL. TIC je jedna z hodnot 1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 16, 20, 24, 30, 40, 60. FX je jeden efekt pro celý pattern: 0 žádný, 1 vibrato, 2 arpeggio, 3 fade out. LEVEL je 0 až 3.

4bitový

Krok je pouze nota. Záhlaví má TIC, LOOP (zapnuto nebo vypnuto), WAVE (stejné čtyři vlny) a OCT (ve které oktávě noty leží). TIC je jedna z hodnot 4, 6, 8, 10, 12, 15, 20, 30.

2bitový

Krok je prázdný, buben, hit nebo crush. Záhlaví má TIC a BASE. TIC je 8, 15, 16 nebo 30. BASE určuje, jak vysoko se nachází buben a hit.

1 bit

Krok je nota. TIC je 8, 15, 16 nebo 30, stejné čtyři rychlosti jako 2bitový kód.

Písně

Karta ZVUK představuje otevřenou skladbu. Osm sloupců představuje kanály. Buňka představuje ID vzoru. Číslo 255, nakreslené prázdné, znamená, že kanál je v daném řádku tichý. Sloupec X představuje příkaz: pokračovat, zastavit, začátek smyčky nebo konec smyčky. Stop ukončí skladbu za řádkem. Konec smyčky skočí zpět na začátek poslední smyčky.

Řada končí, když skončí vzor nejvíce vlevo s LOOP OUT na 0. Vzory dále vpravo danou řadu neprodlužují: delší vzor se zkrátí, kratší vzor začíná znovu od svého prvního kroku, dokud řada neskončí. Vzor s LOOP OUT nad 0 se opakuje mezi LOOP IN a LOOP OUT a řadu nekončí. Pokud se všechny vzory v řadě opakují, řada tam zůstává.

Zkratky

Klíč	Akce
Klávesy klavíru	Napište poznámku ke kroku vybranému myší.
Nahoru / Dolů	Změňte hodnotu pod kurzorem.
Plocha	Přehrajte otevřený SFX nebo otevřenou skladbu z prvního řádku. Spuštění/zastavení nebo restartování v nastavení SFX EDITORU.
Ctrl+Z / Ctrl+Y	Zpět / znovu.
Ctrl+C / Ctrl+X / Ctrl+V	Zkopírujte, vyjměte, vložte výběr.

DATA

Pojmenované tabulky. Každá tabulka se skládá z řádků a sloupců. Z kódu tyto tabulky používají `dget` , `dset` , `dstr` , `drows` a `dcols` . Čísla řádků a sloupců v kódu začínají na 0.

Typy sloupců, cyklicky přepínané v záhlaví:

- **bool** — jeden bit.
- **2** — dva bity.
- **4** — čtyři bity.
- **8** — bajt.
- **16** — 16 bitů, pro ID a větší čísla.
- **spr** — ID spritu.
- **str** — text, vykreslený systémovým fontem.
- **str8** a **str10** — stejný text, nakreslený výše.
- **f16** a **f32** — poloviční přesnost plováku a plný plovák.

Pokud je kurzor umístěn na záhlaví sloupce, stisknutí klávesy Ctrl+posouvání změní typ daného sloupce. Obyčejné posouvání se pohybuje mezi sloupci nebo řádky dolů, podle toho, co vyberete v nabídce NASTAVENÍ → EDITOR DAT. Ctrl tento směr přepíná, a proto Ctrl+posouvání v záhlaví mění typ, když je kolečko nastaveno na sloupce.

Šipky slouží k pohybu. Stisknutím klávesy Enter upravujete buňku. NASTAVENÍ → EDITOR DAT:

- **Kolečko** – posouvání sloupců nebo řádků.
- **Barvy** – mřížka, záhlaví a výběr.

Aktiva

Funkce ASSETS importuje a exportuje otevřenou část, nikoli celý projekt. Sprity používají sprity otevřené v programu PAINT na otevřeném listu. Mapa je otevřená mapa. SFX je otevřený vzor. Hudba je otevřená píseň. Data jsou otevřená tabulka. Výjimkou je kód: každý tabulátor se zapisuje a každý tabulátor se zpětně čte.

Druh	Soubor	Co se pohybuje
Skřítci	`.png`	Otevřený sprite.
Mapa	`.p37map`	Otevřená mapa.
Zvukové efekty	`.p37sfx`	Otevřený vzor.
Hudba	`.p37mus`	Úvodní skladba. Čísla patternů zůstávají tak, jak jsou. Pokud potřebujete tyto zvuky, exportujte je samostatně.
Data	`.p37dat`	Otevřený stůl.
Kód	`.lua`	Každá záložka.

Nastavení

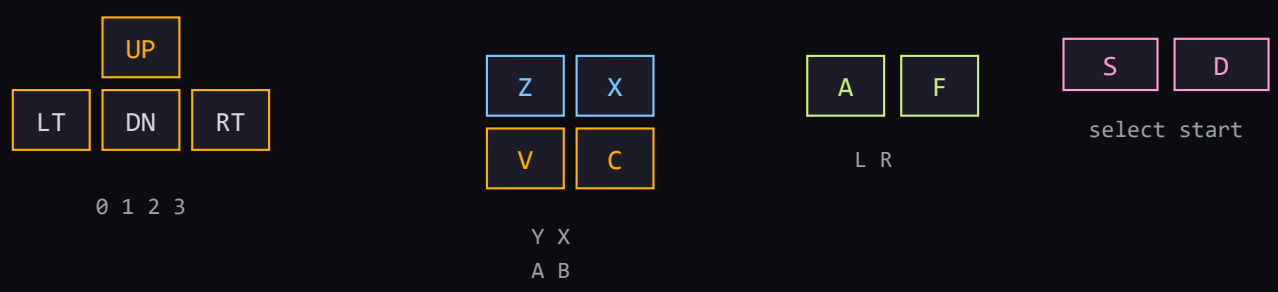
Kořenový seznam:

Strana	Co ukládá
PROFILY	Uložení a načtení všech nastavení. Nové, přejmenování, přepsání, smazání. Zkopírování nebo vložení profilu jako textu.
Menu	Barvy oken, kláves, menu, volitelné pozadí.
HERNÍ PANEL	Náhled a okno přehrávání.

EDITOR KÓDU	Zvýraznění a nápovědy.
EDITOR BARVY	Štětec, paleta, zobrazení mapy.
EDITOR ZNAČKOVÝCH EFEKTŮ	Syntezátor a zadávání not.
EDITOR DAT	Směr a barvy kol.
JAZYK	Jazyk menu a nápovědy.
LICENCE	Licenční texty, včetně písma Fusion Pixel a SDL3.

GLOBÁLNÍ:

- **ROZVRŽENÍ** — uspořádání panelů, výše.
- **HLASITOST** — hlavní úroveň, 0 až 10.
- **VAZBY KLÁVES** — klávesy pro každého z 8 hráčů. Stisknutím klávesy ji zachytíte. Pokyny, ABXYLR, vybrat.
- **Prezentovat** – přeskočit nečinné snímky nebo vykreslit všechny snímky.
- **Celá obrazovka** – bez okrajů.
- **Obnovit** — FPS okna. Krok vlevo a vpravo 60, 90, 120 a max. Enter zadá číslo až do 1000. 0 znamená bez omezení.
- **Vsync** — čekání na zobrazení mezi snímky.
- **Měřítko** — násobitel celočíselného okna.
- **Shadery** — CRT a filtry v editoru, ne ve hře.
- **Barva** — text nabídky a zvýraznění.
- **Pozadí** – volitelný PNG soubor oříznutý na 128 bodů za nabídkou. Volbou Vymazat jej odstraníte.
- **Kurzor editoru** – tvar ukazatele.
- **Myš** – plovoucí ukazatel nebo přichycení k pixelům.
- **Rychlost uživatelského rozhraní** — jak rychle se podržený směr opakuje v seznamu.
- **Průzkumník** – seznamy souborů začínají v kořenovém adresáři počítače nebo ve složce exe.



Výchozí klávesnice. A je potvrzení v menu. B je návrat. Ve hře jsou to tlačítka 4 a 5. Vybrat (S) je tlačítko 10, nikoli zkratka v menu.

Tlačítko	Role	Výchozí klíč
0	Vlevo	Šipka vlevo
1	Právo	Šipka vpravo
2	Nahoru	Šipka nahoru
3	Dolů	Šipka dolů
4	A, potvrďte	PROTI
5	B, zpět	C
6	X	X
7	Y	Z
8	R	F
9	L	A
10	Vybrat	S
11	Start	D

Vytvořit

Vyberte platformu: Windows, Linux, Mac, Android, iOS nebo web.

- **Název** — název souboru sestavení.
- **Autor** – ve Windows a Linuxu. Toto je název složky pro uložené soubory.
- **ID aplikace Steam** — ve Windows a Linuxu. Prázdné pole znamená, že Steamworks není k dispozici. Vyplněné ID mapuje `achieve(0)` … `achieve(15)` na achievementy ve službě Steam ACH_0…ACH_15.
- **Id** — identifikátor balíčku, pokud ho platforma požaduje.
- **Skrýt kód** – na překryvných vrstvách na počítači a mobilních zařízeních. Zpomaluje to `peek` herní kazety. Nejedná se o zámek.
- **Celá obrazovka** – exportovaný přehrávač se spustí v režimu celé obrazovky.
- **Shadery** — až tři vrstvy zapečené do přehrávače.
- **Stáhnout .p37** – Pouze pro web. V nabídce pozastavení se zobrazí položka pro stažení.
- **Ikona** — dlaždice z listu sprite: list, velikost, x, y. Nebo oříznutý obal ve formátu PNG.
- **Exportovat** – vyberte složku. V nabídce se zobrazí zpráva SESTAVENÍ ULOŽENO nebo SESTAVENÍ NEÚSPĚŠNÉ.

Vytvořte sestavení s připojeným internetem. Úvodní text se může anonymně zeptat serveru, zda je znám hash tohoto sestavení. Tato kontrola je zde ze dvou důvodů. Chrání mě, autora PIX37, před kopiemi nástroje, které nebyly moje, abych je mohl rozdávat. Chrání také osobu, která hru vytvořila. Vzít si hru někoho jiného a uvést

ji jako svou vlastní se stává běžnou praxí na trhu s telefony, jako je Play (Android) nebo iOS. Hra se stále spustí. Pokud je počítač offline, úvodní text říká, že hra není připojena. Tato věta je také pro hráče nápovědou, že kopie nemusí pocházet z legálního zdroje. Je to malá kontrola, takže pokud byla vaše hra ukradena a prodána bez vašeho souhlasu někým jiným, můžete to uplatnit v obchodě, jako je Play (Android), Steam nebo itch.io. Server ukládá čas, kdy byl hash poprvé zaregistrován, vedle hashe a později jej nemění. Dřívější čas je původní sestavení. Hash registrovaný po tomto čase je pozdější kopie.

Chci, aby tato kontrola chránila lidi, kteří vytvářejí hry. Engine už za ně dělá spoustu práce a tohle je jedna z nich. Chápu, že to může být kontroverzní. Pokud se komunita rozhodne, že je lepší toto mikro-DRM zrušit, udělám to.

Stará herní kazeta z předchozího editoru se po změně jazyka může odmítnout spustit. Otevřete v tomto editoru projekt `.pix37` a exportujte nový soubor `.p37` .

Připojte se k PIX37.PARTY

PŘIPOJENÍ K PIX37.PARTY je uvnitř aplikace. V prohlížeči není žádná přihlašovací stránka. Přihlašujete se pomocí tiketu Steam nebo přes itch.io pomocí klávesnice a myši. Služba zobrazuje jedinečné zobrazované jméno o délce 2 až 16 znaků. Licenční klíč a Steam ID se ostatním hráčům nezobrazují.

Po splnění podmínek služby se centrum nachází v sekcích Oblíbené, Katalog, Žebříček, Hráči a váš profil.

Vývojářská licence je editor. Vytvoříte hru, nahrajete ji a necháte hvězdičky. Herní licence je hráč. Srdíčka jsou od lidí hrajících na běžeckém hřišti.

- **Katalog** – procházení her. Filtry zahrnují žánr (až tři), velikost obrazovky, velikost herní kazety a to, zda nahrávaný soubor obsahuje projekt `.pix37` .
- **Oblíbené** — hry, které jste označili.
- **Nahrát** — vyžaduje vývojářskou licenci. Odešlete soubor `.p37` a můžete přiložit projekt `.pix37` . Soubor `.p37` je zabalený bajtkód. Je obtížné jej přečíst a lze jej záměrně zneužít. Soubor `.pix37` je soubor projektu, se kterým skutečně pracujete: zdrojový kód, obrázky a zvuk ve formě, kterou může otevřít a učit se z ní jiná osoba. Jeho odeslání umožňuje ostatním tvůrcům studovat hru namísto komprimované herní kazety. Součástí je název, autor, žánry, obal, ikona a názvy achievementů. Odeslané číslo RAM představuje vrchol, kterého herní kazeta dosáhla.
- **Komentáře a achievementy** – na stránce hry. Odemknutí achievementu, když jste přihlášeni, se ukládá na daný účet i na počítač.
- **Pořadí** – vývojáři seřazení podle hvězdiček nebo srdíček. Každý řádek zobrazuje oba počty. Otevřete jméno a zobrazí se hry daného autora seřazené podle srdíček.
- **Hráči** — účty seřazené podle odemčených úspěchů.

Kód

Jazyk je malý skript ve stylu Lua. Tato polovina začíná s prázdným projektem, učí jazyk a poté ukazuje krátký příklad každé funkce.

První program

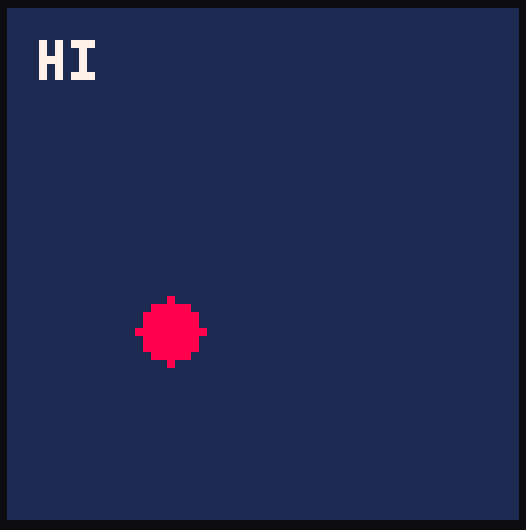
Vytvořte projekt s rozlišením 64×64 a 4bitovou hloubkou barev. Otevřete CODE. Nahraďte tabulátor tímto:

```
_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 ji spustí. Šipky posouvají kruh. Escape hru zastaví a vrátí do editoru.



Stejná tři volání: `cls(1)` vykreslí tmavě modrou obrazovku, `print` vykreslí HI a `circfill` vykreslí růžový kruh.

Co běží a kdy

- `_init` se spustí jednou.
- `_update` běží na 60 Hz. Sem dejte pohyb a tlačítka.
- `_draw` maluje. Bez argumentu fps maluje tak rychle, jak jen počítač a monitor dokážou. `_draw(30)` omezuje malování na 30 fps a ponechává logiku na 60.
- `_set_fps(n)` tento limit později změní. Hodnota 0 jej vynuluje.

Názvy nerozlišují velká a malá písmena; editor je ukládá malými písmeny. `function` a `fun` jsou stejné klíčové slovo.

Jazyk

Základy programování

Program je seznam řádků, které herní kazeta spustí. Každý nadpis nejprve ukazuje tvar řádku a poté malý příklad. Řádek, který začíná znakem `--` , je komentář: poznámka pro vás. Hra ho přeskočí.

Proměnné

Proměnná je název, který obsahuje hodnotu. Název si zvolíte. Pozdější řádky jej mohou číst a uložit novou hodnotu pod stejným názvem.

Tvar je `name = value` . `local name = value` uchovává tento název uvnitř aktuálního bloku, takže ho zbytek herní kazety nevidí. Bez `local` je název globální a vidí ho každá karta.

```
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` znamená, že neexistuje žádná hodnota. `true` a `false` jsou yes a no. Název, který jste nikdy nenastavili, se čte jako `nil` .

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

Stoly

Tabulka obsahuje mnoho hodnot pod jedním názvem. Seznam používá čísla. První položka je 1, ne 0. Záznam používá názvy. Tabulka může obsahovat další tabulku.

Tvar seznamu: `{ value, value }` . Tvar záznamu: `{ name = value, name = value }` . Seznam záznamů je tabulka, jejíž položky jsou tabulky.

```
-- 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
```

Stav a tělo

Podmínka je otázka s odpovědí ano nebo ne, například `hp > 0` nebo `btnd(4)` . Tělo je skupina řádků, které se spustí při vstupu do bloku. `if` a každá smyčka používá stejný tvar: podmínka, poté tělo a nakonec `end` . `repeat` se uzavírá na `until` místo `end` .

Li

Funkce `if` spustí své tělo pouze tehdy, když je podmínka pravdivá. `elseif` zkusí jinou podmínku. `else` spustí, když jsou všechny podmínky nepravdivé.

Tvar: `if condition then` , pak tělo, pak `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
```

Smyčky

Smyčka opakuje tělo smyčky. `while` nejprve kontroluje podmínku a pokračuje, dokud je pravdivá. `repeat` spustí tělo smyčky a poté kontroluje, takže smyčka se vždy spustí alespoň jednou. `for` counts. Poslední číslo je

zahrnuto. Bez třetího čísla je krok 1, takže pokud je začátek již za koncem, smyčka se nespustí. Odpočítávání zpět vyžaduje záporný krok.

```
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
```

Chůze kolem stolu

`all` navštíví každou položku seznamu, od první. `pairs` navštíví každý název a hodnotu uloženou pod tímto názvem.

```
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
```

Funkce

Funkce je pojmenovaný recept. Tvar je `function name(arguments)` , poté tělo a nakonec `end` . Voláte ji napsáním názvu a závorek. `return` odesílá zpět hodnotu a může jich odeslat více. Argumenty jsou lokální hodnoty dané funkce.

```
function greet(who)       -- who is a local that holds what you pass in
  return "hi "..who       -- .. glues two texts. return sends the result back
end

print(greet("ada"))       -- draws hi ada
```

```
function pair()
  return 1, 2             -- two values
end

a, b = pair()             -- a becomes 1, b becomes 2
```

`local function` je funkce, která vidí lokální proměnné kolem sebe. To je uzávěr. Tabulka může pojmout jednu a volá se s tečkou.

```
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
```

Ano, ne a výběr

`not` obrací podmínku. `and` je pravdivá pouze tehdy, když jsou pravdivé obě strany. `or` je pravdivá, když je pravdivá kterákoli ze stran. `a and b or c` vybere `b` když je pravdivé `a` , jinak `c` . `cond ? a : b` je stejná myšlenka v jednom řádku: `a` když je podmínka pravdivá, `b` když není.

```
hp = 2
hit = false
night = false

if hp > 0 and not hit then         -- both parts are true, so the body runs
  print("safe", 1, 1, 7)           -- draws safe
end

label = hp > 0 and "ok" or "out"   -- hp is above 0, so label is ok
print(label, 1, 8, 7)

shade = night ? 1 : 7              -- night is false, so shade is 7
print(shade, 1, 16, 7)
```

Matematika a text

`+ - * /` jsou operace sčítání, odčítání, násobení a dělení. `//` dělí a ponechává celé číslo. `%` je zbytek. `^` je mocnina. `..` slepuje dva texty. `#t` je počet položek v seznamu. Porovnání jsou `==` (stejné), `~=` nebo `!=` (různé) a `< > <= >=` . `x += 1` přičítá 1 k x. Stejný tvar funguje pro `-= *= /= ^= %=` .

```
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
```

Kde žije místní

`local` trvá až do `end` , který uzavírá její blok. `if` / `end` je jejím vlastním blokem. Můžete zmínit proměnnou local nad řádkem, která ji nastavuje, ale dokud tento řádek neskončí, hodnota je `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
```

Skákání

`::label::` označuje řádek. `goto label` na něj skočí, ale pouze uvnitř stejné funkce.

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

Paměť, zkrátka

`peek` and `poke` prohlíží živé pixely prostřednictvím jedné 16bitové adresy. Neuchovávají druhou kopii herní kazety. Adresy `0x5F00` až `0x5FFF` jsou registry: kamera, klip (pouze pro čtení), barva kreslení, vzor výplně, tři přemapování palet a čísla stránek. Bajty na obou stranách této díry, celkem 65280, tvoří okno pixelu.

Když se obrazovka vejde do oněch 65280 bajtů, 64×64 a 128×128 ano a 256×256 ne, obrazovka začíná na adrese 0, řádek po řádku. Zbývá jen list spritu vybraný parametrem `0x5F41` (0 je list 0). Pokud je tento list větší než zbývající místo, `0x5F42` vybere stránku a mapa se v okně nenachází. Pokud se list vejde, na konci se nacházejí buňky mapy z mapy vybrané parametrem `0x5F43` , jeden nebo dva bajty na každou, a `0x5F44` mapu načtou stránky, když se na konci nevejde.

Pokud se obrazovka nevejde na celou obrazovku, celé okno je jedna stránka o 65280 pixelech. `poke(0x5F40, n)` otočí stránku obrazovky. `poke(0x5F41, n)` s n jiným než 0 zobrazí list n a list 0 v tomto režimu nemá žádné zobrazení. `poke(0x5F43, n)` s n jiným než 0 zobrazí mapu n, která vítězí nad listem. Mapa 0 v tomto režimu obsahuje pouze `mget` a `mset` .

`peek` , které nezasáhne pixel nebo registr, vrátí 0. `poke` , které tam nezasáhne, nic nedělá. Příznaky Sprite, SFX a tabulky DATA se v tomto okně nenacházejí. Použijte `fget` , `sfxget` a `dget` .

Praktické využití

Každá funkce je uvedena v dolní části této stránky s praktickým využitím funkcí PIX37. Odkazy pod každým příkladem otevírají další informace o daných funkcích.

 Plynulý pohyb

Herní cartridge o rozměrech 128×128, pohled shora. Sprite 1 je již nakreslený. Šipky zrychlují v libovolném směru a tření vás zpomaluje. Sprite se zalamuje: polovina, která opouští spodní část, je nakreslena nahoře a totéž platí pro ostatní hrany.

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

 Kontrola, zda je další tah blokován

Herní cartridge o rozměrech 64×64. Podlaha a pravá stěna jsou již na mapě, vybarvené spritem 3 a na tomto spritu je nastaven příznak 0. Šipky posouvají sprite 1. Slovo nahoře říká „blokováno“, pokud by další krok překrýval plnou dlaždici, a „uvolněno“, pokud by ne.

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

 Ubrousek s otočením

Stiskněte A. Stírání se vykreslí pomocí funkce flip(), jeden černý řádek po druhém, místo aby se to provádělo uvnitř funkce _draw(). Funkce _draw se během jejího běhu neruší, takže každý řádek zůstává na obrazovce.

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

 Tabulka funkcí (třída)

Sprite 1 je již nakreslený. make_monster funguje jako třída. Vrací tabulku a update a draw jsou metody, které uchovávají souřadnice x a y daného monstra. Číslo nahoře udává, jak daleko monstrum ušlo.

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



Čtyři způsoby, jak přečíst tlačítko

A je tlačítko 4. Podržte ho a náboj stoupá. Uvolněte ho a náboj se vrátí na 0. Podržením opakujte stisknutí tlačítka nahoru a dolů, takže se řádek nabídky neustále pohybuje. btnd je první snímek stisknutí, který tato herní kazeta používá pouze jako začátek daného podržení.

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



Střílejte myší

Sprite 1 je již nakreslený uprostřed. Bílá tečka je kurzor. Kliknutím levým tlačítkem myši se k ní vystřelí pixel. Kolečko mění rychlost, vytištěnou v rohu.

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



Zadejte jméno a potvrďte ho

Stisknutím tlačítka A otevřete pole. Nahoru a dolů měníte blikající písmeno. A ho zapíše, X maže, Y je mezera a B potvrdí. Řádek poté řekne vaše jméno.

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



Kamera na přehrávači

Herní cartridge o rozměrech 128×128. Sprite 1 začíná uprostřed dlaždicové podlahy. Šipky se pohybují. Kamera udržuje sprite vycentrovaný. Osy x a y v rohu se neposouvají, protože camera() bez argumentů vrací pohled zpět pro HUD.

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



Pár řádků vizuálního románu

Řádky jsou v jazyce DATA, angličtině a polštině. A pokračuje, nebo zvolí možnost zůstat, když je na obrazovce možnost. B zvolí možnost odejít, jinak jazyk přepne. X přepne jazyk, když je zobrazena možnost. Dialog používá písmo o velikosti 8px, takže se vykreslují polská písmena.

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



Malování metodou nakukování a pokukování

Herní kazeta o rozměrech 64×64, takže adresa 0 je pixel vlevo nahoře a další řádek začíná na čísle 64. Herní kazeta zapisuje pixely, kopíruje řádek, maže řádek a vytiskne zpět načtené hodnoty.

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



Seznam, který můžete doplnit

A přičte číslo až do 8. B odebere poslední. X odebere první. Seznam se vytiskne dolů na obrazovce.

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



Jméno rozebrané na kusy

Herní kazeta rozdělí ada,12,blue a vytiskne název, jeho první písmeno, 12, a slovo blue.

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



Zahrajte notu a poté ji změňte

Vzor 0 je dlouhý jednu notu a každé přehrání má stejnou hlasitost. A ho zahraje znovu, když daná nota dozní. B zvýší výšku tónu a zahraje danou notu.

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



Píseň napsaná do herní kazety

Dva patterny se smyčkou opakují na řádku 0, jeden na každém z prvních dvou kanálů. Řada zůstává 0. ch0 a ch1 jsou noty, na kterých je každý kanál. Úprava patternu, LEN a LOOP v herní kazetě je způsob, jak zjistit, co tyto ovládací prvky dělají v hudebním panelu.

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



Tři monstra v DATA

Panel DATA již obsahuje sloupec se jménem a sloupec s hp, kde je uveden sliz, netopýr a oko. A slizovi ubližuje. Čísla vpravo udávají jeho hp.

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



Zapamatujte si skóre

A přidá bod a uloží ho. Ukončete hru a znovu otevřete herní kazetu: číslo je stále tam. B na něj zapomene.

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

  Druhá kapitola v jiném souboru

Dvě herní kazety sdílejí úkol v ukládacím slotu. Část 1: A označuje konec hry. Část 2, otevřená jako první, říká, že je třeba dokončit část 1. Poté se otevře část 2 a B smaže uložené místo, aby bylo možné znovu přehrát část 1.

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

 Jeden úspěch

Sprite 8 je již nakreslený. Stiskněte A desetkrát. Upozornění pojmenuje achievement a sprite 8 zůstane, protože je odemčený. B vymaže achievement v lokálním souboru a vynuluje počítadlo.

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

 Místnost, do které můžete vejít

Herní cartridge o rozměrech 256×256. Bílá tečka představuje kurzor. Menu je hostitel, připojit se nebo seznam serverů. Hostitel otevře místnost s názvem chůze a spustí hru. Připojit se posouvá adresu (číslice, tečka, dvojtečka) a B se připojuje. Seznam serverů se posouvá nahoru a dolů a A se připojuje k řádku. Ve hře šipky chodí po spritu 1. Jedna mince je sdílena: hostitel ji pohne a její sebrání přidá 1 ke skóre daného hráče a spustí další. Každý sprite, včetně vašeho, zobrazuje toto skóre vedle sebe. Roh ukazuje, kolik hráčů je ve hře, LAN adresu a veřejnou adresu hostitele. B odchází: vy a ostatní se vrátíte do menu a další hostitel začíná novou místnost se skóre 0.

`btn` `btnd` `mouse` `pset` `spr` `print` `flr` `rnd` `pairs` `nlisten` `nhost` `nannounce` `nlist` `njoin` `nleave` `nsend` `nkb` `nkbstr` `whoami` `nplayers` `nlan` `_mp` `_mp_join` `_mp_leave` `_mp_fail` `palt` `cls` `time` `_init` `_update` `_draw`

 Dialog, který čeká

A zobrazuje další řádek. Funkce se pozastaví na každém yield, dokud znovu nestisknete A.

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

 Počkejte půl vteřiny

Slovo se po zhruba půl sekundy změní z ... na hi. Obrazovka se i po pozastavení korutiny neustále kreslí.

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

 Kruhy, které přicházejí jeden po druhém

Z vrcholu kape pět kruhů, dva rámečky od sebe, a pak spadne.

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

 Dva rámy z hodin

Sprite 1 a sprite 2 jsou dva snímky stejného znaku. Obrázek se střídá čtyřikrát za sekundu.

```
time spr cls flr _draw
```

Každá funkce

Argumenty v hranatých závorkách jsou volitelné. Indexy tabulek a řetězců začínají na 1, s výjimkou řádků a sloupců typu DATA, které začínají na 0. Barvy jsou indexy palet.

Index

_kreslit
_init
_mp
_mp_fail
_mp_připojit se
_mp_leave
_mp_update
_set_fps
_aktualizovat
abs
achclear
achdef
dosáhnout
dosaženo
přidat
všechny
tvrdit
atan2
kapela
bnot
bor
btn
btnd
btnp
btnu
bxor
fotoaparát
strop
chr
cirk
circfill
clickl
clickm
clickr
klip
cls
spoluvytvářet
barva
jádro
cos
costatus
počítat
kurzor
dcols
del
lahůdky
delsave
dget
se utopí
dset
dstr
fget
fillp
převrátit
flr
foreach
fset
info
ipairs
klíček
klíč
keystat
keystr
linka
zatížení
log
lshr
mapa
max
memcpy
memset
mget
střední
min
myš
mset
hudba
musicstat
oznámit
zavolat
další
nhost
připojit se
nkb
nkbstr
nlan
nlist

nlist
poslouchej
poznámka
nplayers
nsend
nstun
obj
oválný
oválná výplň
balení
páry
kamarád
palrgb
palt
pcal
nahlédnout
nahlédnout 2
nahlédnout 4
výkon
pget
šťouchnout
poke2
poke4
tisknout
tisk 10
tisk8
pset
rect
rectfill
rnd
rotl
rotr
správně
rrectfill
uložit
saveslot
obrazovka
rolovat
vybrat
setmap
setpalette
setspr
settex
setturn
sfx
sfxget
sfxhdr
sfxset
sfxstat
sget
sgn
shl
statistiky
shr
hříchh
songget
sada písní
rozdělit
spr
sqrt
srand
sset
sspr
stat
sub
čas
tline
tonum
tostr
rozbalit
whoami
výnos

Životní cyklus

_init

`_init() ... end`

Spustí se jednou při spuštění herní kazety. Kód, který se nachází mimo jakýkoli hook, je také složen do _init.

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

_update

`_update() ... end`

Herní logika. Voláno na 60 Hz, i když je _draw omezeno nebo je hostitel pomalejší. Přečtěte si tlačítka zde.

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

_draw

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

Nakreslí jeden snímek. Bez argumentu nebo s hodnotou 0 se spouští tak často, jak často hostitel zobrazuje. Předejte n (1..999) pro omezení kreslení na n snímků za sekundu. _update zůstává na 60 Hz.

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

_set_fps

`_set_fps(n)`

Mění limit _draw během běhu hry. 0 znamená neomezený limit. n je omezeno na 0..999.

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

flip

`flip()`

Zobrazí obrazovku a pustí se do dalšího snímku. Použijte ji pouze v případě, že kreslíte vlastní smyčku snímků. Nečekejte po ní.

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

time

`time()`

Sekundy od _init.

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

Vstup

btn

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

1, zatímco je stisknuto tlačítko b, jinak 0. p je hráč, 0..7. Bez argumentů vrací bitovou masku hráče 0. Tlačítka: 0 vlevo, 1 vpravo, 2 nahoru, 3 dolů, 4 A, 5 B, 6 X, 7 Y, 8 R, 9 L, 10 vybrat, 11 začít.

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

btnd

`btnd(b, [p])`

Pravda pouze v rámci _update, když je stisknuto tlačítko b.

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

btnp

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

True při stisknutí, poté znovu při podržení: 30 tiků zpoždění, poté každých 8 tiků. Bez tlačítka vrací masku.

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

btnu

`btnu(b, [p])`

Pravda, pouze na rámu se tlačítko uvolní.

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

mouse

`mouse()`

Vrací osy myši x a y v pixelech hry.

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

clickl

`clickl([p])`

Levé tlačítko myši. clickl() je stisknuto. clickl(1) je okraj stisknutí.

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

clickr

`clickr([p])`

Pravé tlačítko myši. Stejná hrana s 1 argumentem jako clickl.

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

clickm

`clickm([p])`

Prostřední tlačítko nebo kliknutí kolečka.

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

scroll

`scroll()`

Pohyb kolečka myši v tomto snímku. Pozitivní a záporné hodnoty závisí na směru otáčení kolečka.

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

keyon

`keyon()`

Otevře textové pole ovládané tlačítky. Stisknutím kláves nahoru a dolů se mění písmeno A–Z. A toto písmeno zapíše. X ho smaže. Y je mezera. B pole potvrdí a zavře. Klávesnice do něj nepíše.

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

keyoff

`keyoff()`

Skryje textové pole.

```
keyoff()
```

keystr

`keystr([s])`

Text v poli, poté 1 v rámečku B potvrzuje, poté se písmeno nahoru a dolů mění. Toto písmeno je prázdné, jakmile se pole zavře. keystr("") vymaže text.

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

keystat

`keystat()`

1, pokud je textové pole otevřené, jinak 0.

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

Výkres

screen

`screen()`

Aktuální velikost obrazovky herní kazety v pixelech. Vrací šířku a poté výšku. Toto je rozlišení hry, nikoli okno editoru.

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

cls

`cls([col])`

Vymaže obrazovku. Výchozí barva je 0.

```
cls(1)
```

color

`color(col)`

Nastaví barvu vykreslování, která se použije, když pozdější volání vynechá hodnotu col. Bez argumentu vrací aktuální barvu.

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

camera

`camera([x, y])`

Provede offset při každém volání draw. camera() se resetuje na 0, 0.

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

clip

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

Omezí kreslení na obdélník. Nenulový pátý argument protíná aktuální klip. clip() obnoví celou obrazovku. Registry klipu jsou pouze pro čtení.

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

cursor

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

Nastaví, kam se má vytisknout další hodnota bez souřadnic, a volitelně i barvu. cursor() vrací x, y.

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

pset

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

Nastaví jeden pixel obrazovky. Počítá se do statistiky px.

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

pget

`pget(x, y)`

Barva pixelu obrazovky v bodech x, y.

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

print

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

Kreslí text o rozměrech 3×5 (UTF-8, oříznutý z písma Fusion). Bez x a y se tiskne v místě kurzoru a přesune se na další řádek.

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

log

`log(...)`

Připojí argumenty oddělené mezerami do textového bufferu. pix37_run vypíše tento buffer na stdout. Nevykresluje.

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

print8

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

Stejné jako tisk, s fontem Fusion o velikosti 8 px.

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

print10

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

Stejné jako tisk, s fontem Fusion o velikosti 10 px.

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

line

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

Řádek od x0,y0 do x1,y1. Dva argumenty pokračují od předchozího konce. line() resetuje tento řetězec.

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

circ

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

Obrys kruhu. Poloměr je v pixelech.

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

circfill

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

Vyplněný kruh.

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

oval

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

Obrys elipsy uvnitř ohraničovacího rámečku. Rohy jsou inkluzivní.

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

ovalfill

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

Vyplněná elipsa v ohraničujícím rámečku.

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

rect

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

Obrys obdélníku. Rohy jsou inkluzivní.

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

rectfill

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

Vyplněný obdélník.

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

rrect

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

Obrys zaobleného obdélníku. x a y jsou levý horní roh. w a h jsou velikosti v pixelech; obě musí být větší než 0. r je poloměr rohu a výchozí hodnota je 0. Je omezen na polovinu kratší strany.

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

rrectfill

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

Vyplněný zaoblený obdélník. Stejné argumenty jako u funkce rrect.

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

fillp

`fillp([p])`

16bitový výplňový vzor pro později vyplněné tvary. 0 je plná. fillp() vrací aktuální vzor. Bity procházejí vzorem 4×4.

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

pal

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

Přemapuje barvu c0 na c1. Režim 0 je paleta kreslení, 1 paleta zobrazení, 2 paleta výplňových vzorů. Funkce pal() resetuje.

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

palrgb

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

Nastaví jednu položku palety na RGB, každý kanál 0..255.

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

palt

`palt([col, t])`

Průhlednost. Funkce palt(col) nebo palt(col, true) způsobí, že sloupec bude průhledný. Funkce palt(col, false) jej učiní plným. Funkce palt() způsobí, že všechny barvy budou neprůhledné.

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

setpalette

`setpalette(n)`

Vybere banku palet n. setpalette() vrací aktuální banku.

```
setpalette(1)
```

tline

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

Nakreslí čáru na obrazovce a zároveň vzorkuje pixely mapy z pozice mapy mx, my. mdx, mdy krok za krokem posune vzorkování mapy podél čáry.

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

Sprity a mapy

spr

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

Kreslí sprite n v bodech x a y. w a h jsou velikosti v pixelech, výchozí hodnota je 8. fx a fy se při nenulové hodnotě otočí horizontálně a vertikálně. tex je karta listu sprite. rot je úhel v otáčkách (0..1), stejný směr jako sin a cos. 0 ponechává sprite neotočený.

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

sspr

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

Zkopíruje obdélník ze sprite sheetu na obrazovku, zmenšený na dw pomocí dh. Výchozí nastavení zachovává velikost zdroje.

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

sget

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

Barva jednoho texelu na archu spriteů.

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

sset

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

Zapíše jeden texel na sprite sheet.

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

fget

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

Příznaky Sprite. fget(n) vrací bajt příznaku. fget(n, f) je true, když je nastaven bit příznaku f (0..7).

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

fset

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

Nastaví příznaky spritu. fset(n, v) zapíše celý bajt. fset(n, f, v) nastaví jeden bit.

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

setspr

`setspr(n)`

Vybere aktivní list sprite. setspr() vrátí aktuální list. Stejná banka jako settex.

```
setspr(1)
```

settex

`settex(n)`

Alias setspr.

```
settex(0)
```

map

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

Kreslí mapu. Buňky cx a cy mapy se vykreslí na obrazovce sx a sy. Hodnoty cw a ch = 0 znamenají, že se vykreslí celá mapa. Vrstva je akceptována a ignorována. tex je list dlaždic.

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

mget

`mget(cx, cy)`

ID spritu v buňce mapy cx, cy.

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

mset

`mset(cx, cy, n)`

Zapíše sprite n do buňky mapy cx, cy.

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

setmap

`setmap(n)`

Vybere mapu n. setmap() vrátí aktuální mapu. Mapa 0 je skutečná mapa.

```
setmap(1)
```

Paměť

peek

`peek(addr)`

Přečte jeden bajt z 64KB okna. Okno je živá obrazovka, poté pixely spritu a poté mapování buněk, když se obrazovka vejde do 65280 bajtů. Větší obrazovky jsou stránkovány pomocí poke(0x5F40, page). Adresy, které nejsou namapovány, vracejí 0. Také se zapisuje @addr.

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

poke

`poke(addr, ...)`

Zapisuje bajty na adresu. Další argumenty zapisují následující bajty. Nenamapované adresy jsou ignorovány. Také se zapisuje $addr, val. Píchnutí do obrazovky se počítá jako zápis pixelu.

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

peek2

`peek2(addr)`

Neznaménová 16bitová hodnota, little-endian, na adrese

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

poke2

`poke2(addr, val)`

Zapíše 16bitovou hodnotu bez znaménka, little-endian.

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

peek4

`peek4(addr)`

Neznaménová 32bitová hodnota, little-endian.

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

poke4

`poke4(addr, val)`

Zapíše 32bitovou hodnotu bez znaménka, little-endian.

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

memcpy

`memcpy(dst, src, len)`

Zkopíruje jen bajtů do stejného okna o velikosti 64 KB.

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

memset

`memset(dest, val, len)`

Naplní jen bajtů počínaje od cíle hodnotou.

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

Matematika

abs

`abs(x)`

Absolutní hodnota.

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

flr

`flr(x)`

Zaokrouhlí směrem k mínus nekonečnu.

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

ceil

`ceil(x)`

Zaokrouhlí směrem k plus nekonečnu.

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

max

`max(a, b)`

Větší z těchto dvou hodnot.

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

min

`min(a, b)`

Menší z těchto dvou hodnot.

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

mid

`mid(a, b, c)`

Prostřední ze tří hodnot. Užitečné jako svorka.

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

sgn

`sgn(x)`

Znaménko x: -1, 0 nebo 1.

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

sqrt

`sqrt(x)`

Druhá odmocnina. Záporný argument není číslo.

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

sin

`sin(x)`

Sinus x je v postupném oběhu: 0, 0,25, 0,5, 0,75, obíhá jednou kružnici. Ne v radiánech.

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

cos

`cos(x)`

Kosinus má zase stejnou konvenci jako sin.

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

atan2

`atan2(dx, dy)`

Úhel vektoru dx, dy se střídavě mění od 0 do 1.

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

rnd

`rnd([x])`

Náhodné číslo v [0, x). Výchozí hodnota x je 1, takže rnd() je 0..1. Pro kostku je to flr(rnd(6))+1.

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

srand

`srand(seed)`

Nastartuje generátor náhodných čísel, aby se pozdější sekvence RND mohla opakovat.

```
srand(1)
```

band

`band(a, b)`

Bitové AND. Také se píše a & b.

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

bor

`bor(a, b)`

Bitový OR. Také se píše a | b.

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

bxor

`bxor(a, b)`

Bitový XOR. Také se píše jako ^^ b.

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

bnot

`bnot(a)`

Bitové NE. Také se píše ~a.

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

shl

`shl(a, n)`

Posun doleva. Také se píše jako << n.

shr

`shr(a, n)`

Aritmetický posun doprava. Také se píše a >> n.

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

lshr

`lshr(a, n)`

Logický posun doprava. Také se píše >>> n.

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

rotl

`rotl(a, n)`

Otočení bitů doleva. Také se píše <<> n.

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

rotr

`rotr(a, n)`

Otočit bity doprava. Také se píše >>< n.

Tabulky a hodnoty

add

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

Připojí v k tabulce t. Pokud je zadáno i, vloží se na index založený na 1.

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

del

`del(t, v)`

Odstraní první položku rovnou v.

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

deli

`deli(t, [i])`

Odstraní index i a vrátí tuto hodnotu. Bez indexu i odstraní poslední položku.

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

all

`all(t)`

Iterátor nad částí pole, v daném pořadí. Díry zastavují procházku stejně jako pole založené na 1.

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

foreach

`foreach(t, fn)`

Volá fn(v) pro každý prvek pole.

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

pairs

`pairs(t)`

Iterátor nad každým klíčem a hodnotou, včetně pojmenovaných polí.

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

ipairs

`ipairs(t)`

Iterátor nad částí pole s indexem založeným na 1.

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

next

`next(t, [k])`

Vrací další klíč a hodnotu po k. next(t) začíná na začátku. Používá se páry.

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

count

`count(t, [v])`

Délka části pole. Pokud je zadáno v, spočítá, kolik položek se rovná v.

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

pack

`pack(...)`

Zabalí argumenty do nové tabulky.

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

unpack

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

Vrací t[i] až t[j] jako samostatné hodnoty. Výchozí hodnoty pokrývají celou část pole.

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

select

`select(i, ...)`

select("#", ...) vrací počet následujících argumentů. select(i, ...) vrací argumenty od i dále.

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

assert

`assert(c, [msg])`

Zastaví se s chybou, když je c false nebo nil. msg nahradí výchozí text.

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

pcall

`pcall(f, ...)`

Volá funkci f. V případě úspěchu vrátí hodnotu true a výsledek. V případě chyby vrátí hodnotu false a text chyby.

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

Struny

chr

`chr(n)`

Jeden znak z bajtového kódu 0..255.

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

ord

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

Bajtové kódy s od indexu i do j. Indexy jsou založeny na 1. Výchozí hodnota je první znak.

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

sub

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

Podřetězec od i do j, založený na 1. Vynechte j pro zachycení ocasu.

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

split

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

Rozdělí s do tabulky. sep má výchozí hodnotu čárka. Číslo sep je šířka bloku. conv převádí části s tonum, pokud je to pravda.

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

tonum

`tonum(s, [base])`

Analyzuje číslo. Základ může být 2 až 36. Vrací nil, pokud text není číslo.

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

tostr

`tostr(v, [digits])`

Převede hodnotu na text. Digits nastaví desetinná místa pro čísla.

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

Zvuk

sfx

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

Přehraje vzor n a vrátí kanál. Vynecháte-li ch nebo předáte -1, vybere se první volný kanál. ch -2 zastaví vzor n na každém kanálu. n -1 zastaví kanál nebo každý kanál, pokud je ch vynecháno. n -2 umožní dokončení smyčky vzoru namísto jejího opakování. sfx() bez argumentů vrátí -1.

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

sfxget

`sfxget(n, step)`

Vrátí tabulku s jedním krokem patternu: [1] výška tónu, [2] vlna, [3] hlasitost, [4] efekt. Výška tónu je 0..63 (C-0 až D#5). Vlna je 0..15. Hlasitost je 0..7 a 0 je tichý zvuk. Efekt je 0..7.

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

sfxset

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

Zapíše jeden krok vzoru. Všech šest argumentů je povinných.

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

sfxhdr

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

Záhlaví vzoru. sfxhdr(n) vrací tabulku záhlaví. sfxhdr(n, key, val) nastaví jedno pole: speed, loop_in, loop_out, level, noise, crush, dtune, rverb, dampn, swing, steps. Čísla 0..10 vybírají stejná pole v tomto pořadí. speed je TIC, tiky na krok. Vyšší je pomalejší. 15 je 120 úderů za minutu. 16bitový systém akceptuje 1..255. Užší syntezátory se přichytí k jejich TIC seznamu.

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

sfxstat

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

ID vzoru přehrávané na kanálu ch nebo -1. Výchozí kanál je 0.

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

note

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

Hraje jednu notu bez patternu. Výška tónu 0..63, vlna 0..15, hlasitost 0..7, efekt 0..7. Ticky udávají, jak dlouho hraje, ve stejných 120Hz tiketech jako TIC. Vynecháte-li ticky, hraje 15, což je jeden výchozí krok. Efekt běží po celou dobu a poté se kanál ztiší.

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

music

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

Spustí skladbu n shora. music(-1) zastaví. Volání music s ID, které se již přehrává, ji ponechá běžící. music() vrátí ID přehrávané skladby neboli -1. Fade je v milisekundách: skladba se postupně zrychluje nebo zesiluje, když je n -1. Nastavené bity masky představují kanály použité pro skladbu. 0 znamená všech osm. Ostatní kanály zůstávají volné pro zvukové efekty.

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

musicstat

```
musicstat()
```

Tabulka přehrávané skladby. song a row jsou ID skladby a řádek, nebo -1, pokud se nic nehraje. step a pos mají osm čísel. step[1] je kanál 0, step[8] je kanál 7. step je index noty, nebo -1, když je kanál tichý. pos je, jak daleko se nota posunula, od 0 do 1. tick je, kolikrát se tento řádek přehrál. played je, kolik řádků začalo od music(), včetně toho, který se právě přehrává.

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

songget

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

ID vzoru uložené v buňce skladby. ch je 0..7. 255 znamená, že buňka je prázdná. Toto je aktivní volání; text nápovědy kódu stále zobrazuje starší pořadí argumentů.

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

songset

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

Zapíše ID vzoru do buňky skladby. Pro vymazání buňky použijte hodnotu 255.

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

stat

```
stat(n)
```

Pouze zvuk. 16..23 je ID sfx na kanálu n-16. 24..31 je hudební vzor na kanálu n-24. Jakékoli jiné n vrací 0. Časování snímků a RAM jsou výkon a informace, nikoli statistiky.

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

Datové tabulky

dget

`dget(t, row, col)`

Přečte buňku datové tabulky t z panelu DATA. Řádek a sloupec začínají na 0.

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

dset

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

Zapíše do buňky. Pro textový sloupec použijte řetězec.

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

dstr

`dstr(t, row, col)`

Buňka jako text. Čísla jsou převedena.

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

drows

`drows(t)`

Kolik řádků má tabulka t.

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

dcols

`dcols(t)`

Kolik sloupců má tabulka t.

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

Uložení a dosažené úspěchy

save

`save(k, v)`

Uloží v pod klíč k v souboru pro uložení. v může být číslo, řetězec nebo tabulka těchto čísel. Soubor je komprimovaný.

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

load

`load(k)`

Přečte klíč k. Vrátí nil, pokud klíč chybí.

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

delsave

`delsave(k)`

Smaže klávesu k z uložené paměti.

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

saveslot

`saveslot([name])`

Vybere soubor pro uložení ve složce autora. Pokud není uveden název, slot je název hry. Vrátí 1, pokud daný soubor již existuje.

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

achieve

`achieve(i)`

Odemkne achievement i, od 0 do 15. Ukládá se lokálně. Sestavení na platformě Steam také nastaví ACH_i. Relace katalogu jej nahlásí na účet hráče.

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

achieved

`achieved(i)`

1, když je již odemčen achievement i.

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

achdef

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

Upozorňující text pro dosažení i. úspěchu, zobrazený po dobu 15 sekund od spodní části obrazovky. spr je volitelná dlaždice 8×8, tex její list.

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

achclear

`achclear(i)`

Vymaže achievement i v lokálním souboru. Již nastavený achievement Steam ACH_i zůstane nastavený.

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

Síť

whoami

`whoami()`

Vaše ID hráče je 0..15, zatímco je relace spuštěna. Offline je -1.

```
id = whoami()
```

nhost

`nhost()`

1, pokud je tento počítač hostitelem, jinak 0.

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

nplayers

`nplayers()`

Kolik živých ID existuje v 0..15.

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

nlisten

`nlisten()`

Spustí hostování UDP a vrátí LAN adresu jako ip:port.

```
addr = nlisten()
```

njoin

`njoin(addr)`

Připojuje se k IP nebo IP:port.

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

nleave

`nleave()`

Opouští relaci. Vzájemní partneři vidí _mp_leave s argumentem why 0.

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

nsend

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

Odešle ztrátovou zprávu. Název je událost. Následují až čtyři další hodnoty. p od 0 do 15 šeptá danému protějšku; vynechejte p pro odeslání všem.

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

ncall

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

Stejný tvar jako nsend, ale zpráva se opakuje, dokud není potvrzena. Tři neúspěchy za sebou ukončí relaci.

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

nlan

`nlan()`

Aktuální IP adresa LAN po nlisten.

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

nstun

`nstun()`

Vaše veřejná IP adresa:port ze STUN. Host z jiné sítě se k ní může pokusit připojit.

```
pub = nstun()
```

setturn

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

Záložní přepínač TURN selže, když přímá cesta selže. Funkce setturn() s prázdnými argumenty jej vypne. Adresa je host:port.

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

nlist

`nlist(game)`

Tabulka otevřených relací pro daný název hry. Každá položka má adresu, hráče a jméno.

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

nannounce

`nannounce(game)`

Prezenční signál, takže se tento hostitel zobrazí v seznamu hostů pro hru. Zapněte časovač během hostování.

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

nkb

`nkb([on])`

Adresa pole. nkb(1) jej otevře, nkb(0) ho skryje, nkb() vám řekne, zda je otevřené. Nahoru a dolů mění znak, 0–9 nebo tečku nebo dvojtečku. A ho zapíše. X ho smaže. Klávesnice do něj nepíše.

```
nkb(1)
```

nkbstr

`nkbstr([s])`

Text v adresním poli, poté se mění znak nahoru a dolů. nkbstr("") vymaže text.

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

_mp

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

Tuto funkci napíšete. Spustí se, když dorazí zpráva. from je ID odesílatele. name je událost. Následují až čtyři hodnoty.

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

_mp_join

`_mp_join(p)`

Napíšete tuto funkci. Peer obdržel ID p. Ta se pro vás také spustí; whoami() je toto ID.

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

_mp_leave

`_mp_leave(p, why)`

Napíšete tuto funkci. Partner p odešel. Proč 0 je normální odchod. Proč 1 je ztracené spojení.

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

_mp_update

`_mp_update()`

Tuto funkci napíšete. Spustí se každých 15 snímků _update, přibližně 4 Hz. Dobré místo pro odesílání pozic pomocí nsend.

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

_mp_fail

`_mp_fail()`

Napíšete tuto funkci. Relace je nefunkční: hostitel zmizel nebo ncall selhal třikrát.

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

Korutiny

cocreate

`cocreate(f)`

Vytvoří korutinu z funkce f. Nespustí se, dokud není provedena korutina.

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

coresume

`coresume(c, ...)`

Spouští korutinu c, dokud nevrátí hodnotu nebo nevrátí hodnotu. V případě úspěchu: true a vrátené hodnoty. V případě chyby: false a chyba.

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

costatus

`costatus(c)`

„pozastaveno“, „běží“ nebo „mrtvé“.

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

yield

`yield(...)`

Vrací kontrolu zpět tomu, kdo zavolal coresume. Hodnoty se stávají výsledky tohoto coresume.

```
yield(1)
```

Opatření

info

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

Vykreslí živý HUD v _draw: fps (průměr 1 sekunda), upd, drw, px, max, ram, peak, ramc a net, když je relace spuštěná. info() používá tiskový kurzor. info(col) nastavuje pouze barvu. info(x, y) nebo info(x, y, col) ji umístí. Čísla používají B, k nebo m, aby se vešla na malou obrazovku. Samotné volání se nepočítá v px.

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

showstats

`showstats([on])`

Stejný HUD jako v info, vykresluje se automaticky na 0, 0 po _draw, ignoruje kameru a klip. showstats(0) ho skryje. showstats(1) nebo showstats() ho zobrazí.

```
showstats(1)
```

perf

`perf(n)`

Jedno číslo HUD, abyste si ho mohli nakreslit sami. 0 fps, 1 upd, 2 draw, 3 px, 4 max (nejtěžší snímek v tuto sekundu), 5 ram, 6 ramc (ram plus zabalená herní kazeta), 7 peak ram od začátku herní kazety.

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