PIX37 Kılavuzu

Önce editör, sonra kod. Kitabın çevrimdışı kopyası pix37.party/docs adresinde.

Editör

PIX37 DEV editördür. Burada oyunu çizebilir, yazabilir ve seslendirebilirsiniz, ardından bir oyun kartuşu veya bağımsız bir yapı olarak dışa aktarabilirsiniz. Ayrıca bir oyun kartuşunu oynatabilir ve oyun kartuşlarını indirmek ve kataloğu kullanmak için PIX37.PARTY'yi açabilirsiniz. PIX37 DEV EDUCATIONAL aynı editördür ve ücretsizdir. Yapı oluşturma, oyun kartuşu dışa aktarma ve PIX37.PARTY'ye erişim özelliği yoktur. PIX37 Console ise oynatıcıdır. `.p37` oyun kartuşlarını oynatır ve oyun kartuşlarını indirmek ve kataloğa göz atmak için PIX37.PARTY'yi açarsınız.

Projeler ve oyun kartuşları

`.pix37` dosyası projeyi temsil eder: kaynak kod sekmeleri, sprite sayfaları, haritalar, ses ve veri tabloları. `.p37` dosyası ise oyuncunun çalıştırdığı oyun kartuşunu temsil eder: sıkıca paketlenmiş bayt kodu.

Kayıt dosyaları ve başarılar, uygulama verilerinin yanındaki yazar klasöründe bulunur. Windows'ta bu, kayıt dosyası için `C:\\Users\\Ada\\AppData\\Roaming\\PIX37\\Mira\\Pong.p37s`, başarılar için ise `C:\\Users\\Ada\\AppData\\Roaming\\PIX37\\Mira\\Pong.ach` şeklinde görünür. Linux'ta aynı dosyalar `~/.local/share/PIX37/Mira/Pong.p37s` ve `~/.local/share/PIX37/Mira/Pong.ach`'dir. Ada, bilgisayardaki hesaptır. Mira, yazar adıdır ve bu ad klasörün adıdır. Pong, varsayılan olarak oyun başlığına ayarlanmış olan kayıt yuvasıdır. `save`, `.p37s`'yi yazar. Kilidi açılan ilk başarı `.ach`'yi yazar. Dosyalar zaten oradaysa, oyun kartuşu başlatıldığında çalıştırıcı bunları okur.

Geliştirici menüsü

Escape tuşu geliştirici menüsünü açar. Oyun çalışırken Escape tuşuna basmak editöre geri döner. Enter veya A tuşuyla (varsayılan olarak V) onaylayın. Geri tuşu B tuşudur (varsayılan olarak C).

Hiçbir proje yüklenmediğinde menü şu şekildedir: PROJE OLUŞTUR, PROJE YÜKLE, DERLE, OYUN KARTUŞUNU YÜKLE, PIX37.PARTY'YE BAĞLAN, AYARLAR.

Proje yüklendikten sonra izlenecek adımlar şunlardır: PROJEYİ KAYDET, PROJEYİ YÜKLE, PROJE OLUŞTUR, VARLIKLAR, DERLEME, OYUN KARTUŞUNU YÜKLE, PIX37.PARTY'YE BAĞLAN, AYARLAR.

Ctrl+S açık projeyi kaydeder. Ctrl+R ise projeyi çalıştırır.

Bir proje oluşturun

PROJE OLUŞTUR komutu, projeyi oluşturmaya başlamadan önce oyun kartuşunu istiyor.

- **VİDEO** — 1-BİT (2 renk), 2-BİT (4), 4-BİT (16), 8-BİT (256).
- **SES** — sentezleyicinin örnekleme derinliği: 16, 8, 4, 2 veya 1 bit.
- **EKRAN** — ön ayarlar arasında 64×64, 128×128, 160×90, 256×256 ve 320×180 bulunur. ÖZEL boyut, 1×1'den 512×512'ye kadar herhangi bir boyutu seçmenize olanak tanır.
- **PALETLER** — renk bankaları. İlk banka aşağıdaki 16 renkli setle başlar. PALET YÜKLE seçeneği, yerleşik bir seti uygular: PICO-8 16, PICO-8 32, Game Boy, 1-bit, CGA, Sweetie-16 ve Lospec'ten alınan daha uzun bir liste (Resurrect 64, Apollo, Endesga 32 ve 64, CC-29, SLSO8, Pear36, Oil 6, Duel, AAP-64, Lost Century ve diğerleri). Ayrıca bir palet dosyası yükleyebilir veya kendi PNG dosyanızı yükleyip paleti o görüntüden alabilirsiniz.

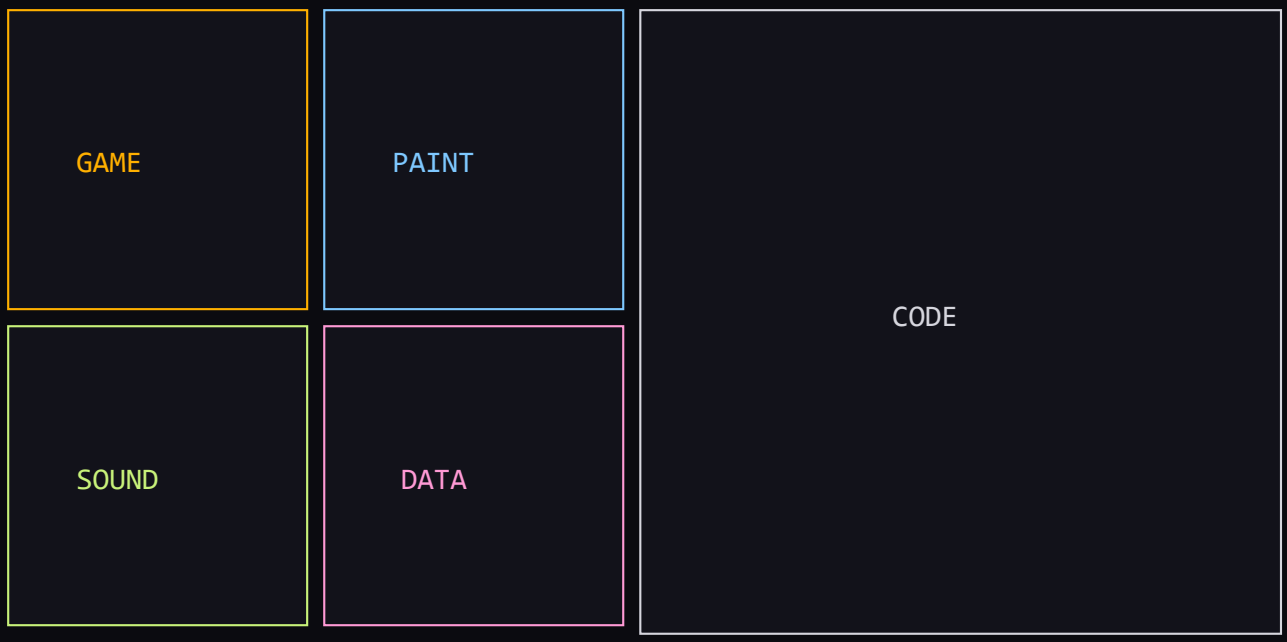


lette — black, navy, purple, green, brown, gray, silver, white, red, orange, yellow, lime, blue, indigo, p

`cls`, `pset`, `spr` ve diğerleri tarafından kullanılan renk indeksleri. Palet düzenleyicisinde herhangi bir indeksi yeniden boyayabilirsiniz.

Beş panel

Editör sırasıyla OYUN, BOYAMA, KOD, SES ve VERİ'den oluşur. Harita BOYAMA içinde yer alır. Ses efektleri ve şarkılar SES içinde bulunur. Çalışmak istediğiniz panele tıklayın. KOD odak noktasındayken, SES ve VERİ harf tuşlarını yok sayar, bu nedenle yazmak piyano çalmayı da engeller.



Tam düzen, `1:1:4:1:1` olarak yazılmıştır. Her sayı, o panelin satırda ne kadar yer kapladığını gösterir: OYUN, BOYAMA, KOD, SES, VERİ. Küçük bir ekran, 128×128'lik bir hücrenin ortasına yerleştirilir. 128 pikselden daha büyük bir ekran, OYUN sütununu genişletir.

AYARLAR → GENEL → DÜZEN, düzenlemeler arasında geçiş yapar:

Kod	Gördüğünüz şey
`1:1:4:1:1`	Her şey. OYUN ve SES üst üste, BOYA ve VERİ üst üste, KOD sağda.
`1:1:4:1:1 2X`	Aynı düzenleme. KOD iki kat daha yoğun çizildiğinden, iki kat daha fazla metin sığar.
`1:0:1:0:1`	OYUN, VERİ, KOD.
`1:0:1:0:1 2X`	Aynı düzenleme, ancak daha yoğun kod ile.
`0:1:1:1:1`	Oyun önizlemesi yok. BOYA, KOD, SES, VERİ.
`0:1:1:1:1 2X`	Aynı düzenleme, ancak daha yoğun kod ile.
`0:1:4:1:2`	Stüdyo. Geniş KOD, daha uzun bir VERİ şeridi, BOYA ve SES.
`0:1:4:1:2 2X`	Aynı düzenleme, ancak daha yoğun kod ile.
`0:0:4:0:4`	Yalnızca KOD ve VERİ, ikisi de büyük boyutlu.
`0:0:4:0:4 2X`	Aynı düzenleme, ancak daha yoğun kod ile.
`0:1:4:1:0`	BOYA, SES ve geniş bir KOD.

`0:1:4:1:0 2X`	Aynı düzenleme, ancak daha yoğun kod ile.
`0:4:0:0:0`	Sadece boya, 256×256, kağıdın bir pikseli tuvalin bir pikseline denk gelecek şekilde.

`2X` satırı aynı panelleri korur. CODE paneli iki kat çözünürlükte çizilir ve aynı hücreye küçültülür, böylece metin, sekmeler ve boşluk yarı yüksekliğinde olur ve ekrana iki kat daha fazla kaynak sığar.

OYUN

Oynatılmakta olan oyun kartuşunun önizlemesi. Bu önizlemeyi çalıştırmak için OYUN paneline tıklayın veya Ctrl+R tuşlarına basın. AYARLAR → OYUN PANELİ bölümünden panel yapılandırılır:

- **Önizleme** — bekleme rengi, sürekli veya düzenleme yaparken otomatik yeniden yükleme.
- **Yeni bir pencerede oynayın** — oyun editörün yanında açılır.
- **Ekran** — önizleme RGB888 veya RGB565 olarak göster.
- **OYUN 64** — 64x64'lük bir ekran ya hücrenin ortasına 1:1 oranında yerleştirilir ya da hücreyi dolduracak şekilde 2 kat ölçeklendirilir.
- **Gölgelendiriciler** — Oyun içindeki CRT ve filtreler, düzenleyicideki filtrelerden ayrıdır.
- **Renk** — panelin arka planı.

BOYAMAK

Sprite sayfaları ve aynı panelin harita tarafında karo haritalar bulunur. Sayfalar ekrandan daha büyük olabilir. Sprite önizleme boyutları 8, 16, 32, 64 ve 128'dir (128 yalnızca sayfa en az 128 piksel olduğunda geçerlidir). Sprite önizlemesini yakınlaştırmak için fare tekerleğini üzerinde kaydırın. Fırça boyutları 1, 2, 3 ve 4'tür.

Anahtar	Alet
B	Kalem
M	Dikdörtgen seçimi
Q	Kement
S	Damga vurmak
G	Kova
BEN	Damlalık
V	Taşınmak
L / Shift+L	Çizgi / eğri
U / Shift+U	İçi boş dikdörtgen / içi boş elips

F / Shift+F	Dolu dikdörtgen / dolu elips
D	Titreklik
[ ]	Daha küçük / daha büyük fırça
Ctrl+Z / Ctrl+Y	Geri al / yinele. Ctrl+Shift+Z tuşlarına basmak da yineleme işlemini yapar.
Ctrl+C / Ctrl+X / Ctrl+V	Kopyala, kes, yapıştır
Ctrl+A	Tuvali seçin

Harita, daha az araçla aynı fikri kullanıyor: B kalem, M dikdörtgen seçim, S damga, V taşıma ve Ctrl+Z, Ctrl+Y, Ctrl+C, Ctrl+V. Harita dokusunu yakınlaştırmak için Alt tuşunu basılı tutun. Haritadaki fırça boyutu, boyama araçlarıyla aynı kalır.

Her bir sprite'ın sekiz bayrak biti vardır. Paint bunları sprite'a yerleştirir; `fget` ve `fset` ise oyunda bunları okur.

AYARLAR → BOYAMA DÜZENLEYİCİ bölümünde bu panel yapılandırılır:

- **Palette** — hangi bankanın çekileceği, ayrıca yükleme ve kaydetme işlemleri.
- **Önizleme** — RGB888 veya RGB565.
- **Fırça imleci** — boyama sırasında kullanılan işaretçi.
- **Piksel yakalama** — fırçanın piksellere nasıl yapıştığı.
- **Büyük fırça** — büyük bir fırça, sol üst köşesinden veya tüm ayak izinden kopar.
- **Soğan zarı** — diğer karelerin bir hayaleti ve o hayaletin hangi yöne baktığı.
- **Harita yakınlaştırma** — gerçek pikseller veya serbest yakınlaştırma.
- **Renkler** — boyama arayüzünün renkleri.

KOD

Sekmeler, proje içindeki ayrı dosyalardır. Bir sekmeyi yeniden adlandırmak için üzerine tıklayın; Enter tuşu onaylamak için kullanılır. Her biri `_init`, `_update` veya `_draw`'yi tanımlayan birkaç sekme, sekme sırasına göre bir araya getirilir: ilk sekme önce çalışır, son sekme en son çalışır. Bir kancanın içinde olmayan kod, `_init`'nin bir parçası olur. Hiçbir kancası olmayan bir oyun kartuşu, `_draw` olarak ele alınır.

Anahtar	Aksiyon
Oklar	İmleci hareket ettirin. Shift tuşu seçimi genişletir.
Ctrl+Sol / Sağ	Sözüyle.
Ana Sayfa / Son, Ctrl+Ana Sayfa / Son	Satırın veya sekmenin başlangıcı veya sonu.

Sekme	İki boşluk bırakın. Tamamlama listesinde adı kabul edin.
Girmek	Yeni satır, önceki gibi girintili. Tamamlama isteğini kabul eder.
Ctrl+C / X / V	Kopyala, kes, yapıştır.
Ctrl+Z / Ctrl+Y	Geri al / yinele.
Ctrl+A	Sekmeyi seçin.
Ctrl+D	Satırı veya seçimi aşağı doğru çoğaltın.
Ctrl+/	`--` ile yorum satırı haline getirin veya yorum satırını kaldırın.
Ctrl+F	Bul. F3 bir sonraki tuşa atlar. Escape tuşu çubuğu kapatır.
Ctrl+G	Bir satır numarasına gidin.
Kaçmak	Öncelikle arama, gitme veya ipucu listesini kapatır. İkinci bir Escape tuşu ise düzenleyiciden çıkarak geliştirici menüsüne geçmenizi sağlar.

Fonksiyon adını yazarken ipuçları açılır. Enter veya Tab tuşuna basıldığında ad eklenir ve ayarlarda TAMAMLANDI ( ) açıksa `(` eklenir. AYARLAR → KOD DÜZENLEYİCİ:

- **Arama** — Etkin sekmede veya tüm sekmelerde Ctrl+F tuş kombinasyonunu kullanın.
- **Yüksek metin** — karakterleri 5 piksele kırpın veya karakterin tamamını gösterin.
- **Dize boyutu** — dize sabitlerinin yazı tipi boyutu.
- **Yorum yazı tipi** — 3×5 veya mevcut kullanıcı arayüzü dilinin Fusion yazı tipi.
- **Tamamla (** — tamamlama türlerinin parantez içinde olup olmadığı.
- **Kod ipuçları** — açık veya kapalı.
- **İpucu dili** — menü dili veya İngilizce.
- **İmleç boşluğu** — boş satırlarda ek kaydırma.
- **Oluk** — satır numaraları.
- **Renk** — arka plan, yorumlar, sekmeler, sayılar, operatörler, anahtar kelimeler, dizeler, motor fonksiyonları, sizin fonksiyonlarınız, değişkenler, parantezler, imleç, seçim, hata satırı, uyarı.

SES

Panelde iki sekme bulunur. SFX bir kalıptır: kısa bir nota dizisi. SOUND ise bir şarkıdır: bu kalıpları sekiz kanala yerleştirir. TIC, bir adımın kaç saniye beklediğini gösterir. Saat saniyede 120 vuruş yapar ve bir adım onaltılık notadır, bu nedenle 15 vuruş dakikada 120 vuruş demektir. Daha yüksek bir TIC, daha yavaş bir kalıptır. Bir TIC diğerini böldüğünde iki kalıp aynı anda devam eder: 15 ve 30 aynı anda devam eder, 15 ve 8 ise etmez.

Piyano, fareyle bir nota seçildiğinde çalmaya başlar. Beyaz tuşlar ZXCVBNM (CDEFGAB) şeklindedir. Siyah tuşlar SDGHJ (C# D# F# G# A#) şeklindedir. Herhangi bir değerin üzerine gelip bir sayı yazın veya üzerine

gelip tekerleği kaydırın. Koddaki perde numaraları 0'dan (C-0) 63'e (D#5) kadar uzanır.

Bir kalıpta, LOOP ve LEN aynı iki sayıdır. 0'ın üzerindeki OUT, IN'den OUT'a kadar döngü yapar ve başlık LOOP olarak okunur. 0'da OUT ve 0'ın üzerindeki IN, LEN'dir: o kadar nota baştan çalınır, ardından kalıp biter ve başlık LEN olarak okunur. Her ikisi de 0'da her adımda çalınır.

Sentezleyicinin ne kadarını düzenleyebileceğiniz, oyun kartuşu için seçilen bit derinliğine bağlıdır. AYARLAR → SFX DÜZENLEYİCİ, bu derinliği, bir notanın otomatik olarak ilerleyip ilerlemediğini (kapalı, nota zinciri veya alan zinciri), Boşluk tuşunun oynatmayı yeniden başlatıp başlatmadığını veya başlatma ve durdurma arasında geçiş yapıp yapmadığını ve düzenleyici renklerini ayarlar.

16 bit

Her adımda bir nota, bir ses seviyesi, bir dalga ve bir efekt bulunur. Ses seviyesi 0 ile 7 arasındadır. 0 sessizliği temsil eder.

Dalgalar:

- **0 üçgen**
- **1 eğimli testere**
- **2 testere**
- **3 kare**
- **4. darbe** — dar bir darbe
- **5 organ**
- **6 gürültü**
- **7 fazör**
- **8 PWM** — genişliği değişen bir kare
- **9 nabız %25**
- **10 sinüs**
- **11 metalik gürültü**
- **12 FM zil sesi**
- **13 koparma**
- **14 süper testere**
- **15 PWM (C64)**

Tek bir nota üzerindeki etkiler:

- **0 yok**
- **1. kaydırma** — bu adımda perde ve ses seviyesi bir sonraki notaya doğru kayar.
- **2 vibrato** — perde dalgalanmaları
- **3. düşüş** — ses sessizliğe bürünüyor, her zamanki vuruş
- **4 kademeli olarak giriş**
- **5 kademeli olarak kaybolma**

- **6 arpej** — bu notayı ve sonraki üç notayı her 8 vuruşta bir tekrarlar. 8. vuruş veya daha aşağıda, her 4 vuruşta bir.
- **7 hızlı arpej** — aynı dört nota, her 4 vuruşta bir veya TIC 8 veya daha düşük olduğunda her 2 vuruşta bir.

Desen başlığı, notların üstünde:

- **TIC** — adım başına vuruş sayısı, 1 ile 255 arasında. 15, dakikada 120 vuruş demektir. Sayıyı yazın veya birer birer artırın.
- **DÖNGÜ İÇERİ / ÇIKIŞ** — ÇIKIŞ değeri 0'ın üzerinde olduğunda, desen İÇERİ'den ÇIKIŞ'a doğru döngü yapar. ÇIKIŞ değerini 0'da bırakırsanız, başlık LEN olur: İÇERİK kaç notanın çalındığını, 0 ise tüm deseni ifade eder.
- **GÜRÜLTÜ** — tüm desen gürültüye dönüşüyor.
- **CRUSH** — her dalganın heyecan verici bir varyantı.
- **DTUNE** — 0 kapalı, 1 hafif bir nakarat, 2 güçlü bir nakarat.
- **RVERB** — yankı: 0 kapalı, 1 kısa, 2 uzun.
- **DAMPN** — alçak geçiş filtresi: 0 kapalı, 1 hafif, 2 güçlü.
- **SEVİYE** — tüm desenin hacmi, 0 ile 7 arası.
- **SWING** — adım adım ilerleme. Tek adımlar biraz daha uzun, çift adımlar biraz daha kısa bekleyin.
- **ADIM SAYISI** — desenin kullandığı adım sayısı: 32, 24, 16 veya 12.

8 bit

Bir adımda bir nota ve bir dalga bulunur. Dört dalga kare, üçgen, testere ve gürültüdür. Başlıkta TIC, LOOP IN/OUT, FX ve LEVEL bulunur. TIC, 1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 16, 20, 24, 30, 40, 60 değerlerinden biridir. FX, tüm kalıp için tek bir efekttir: 0 yok, 1 vibrato, 2 arpej, 3 kademeli azalma. LEVEL ise 0 ile 3 arasındadır.

4 bit

Bir adım sadece bir notadır. Başlıkta TIC, LOOP (açık veya kapalı), WAVE (aynı dört dalga) ve OCT (notaların hangi oktavda yer aldığı) bulunur. TIC, 4, 6, 8, 10, 12, 15, 20, 30'dan biridir.

2 bit

Bir adım boş, bir davul, bir vuruş veya bir ezme olabilir. Başlıkta TIC ve BASE bulunur. TIC 8, 15, 16 veya 30'dur. BASE, davulun ve vuruşun ne kadar yüksekte duracağını belirler.

1 bit

Bir adım bir nota anlamına gelir. TIC, 2 bitlik sistemdekiyle aynı dört hız olan 8, 15, 16 veya 30'dur.

Şarkılar

SES sekmesi açık olan şarkıyı gösterir. Sekiz sütun kanalları temsil eder. Bir hücre bir desen kimliğidir. Boş olarak çizilen 255, o satırda kanalın sessiz olduğu anlamına gelir. X sütunu bir komuttur: devam et, durdur, döngü başlat veya döngü bitir. Durdur, şarkıyı satırdan sonra sonlandırır. Döngü bitir, son döngü başlangıcına geri döner.

Bir sıra, 0 konumunda LOOP OUT bulunan en soldaki desen tamamlandığında sona erer. Daha sağdaki desenler o sırayı uzatmaz: daha uzun olan kesilir, daha kısa olan ilk adımından tekrar başlar ve sıra bitene kadar bu işlem devam eder. 0'ın üzerinde LOOP OUT bulunan bir desen, LOOP IN ve LOOP OUT arasında ilmek yapar ve sırayı sonlandırmaz. Sıradaki her desen ilmek yaparsa, sıra olduğu gibi kalır.

Kısayollar

Anahtar	Aksiyon
Piyano tuşları	Fareyle seçilen adıma bir not yazın.
Yukarı / Aşağı	İmlecin altındaki değeri değiştirin.
Uzay	İlk satırdaki açık SFX'i veya açık şarkıyı çalın. SFX DÜZENLEYİCİ ayarlarından başlatın/durdurun veya yeniden başlatın.
Ctrl+Z / Ctrl+Y	Geri al / yinele.
Ctrl+C / Ctrl+X / Ctrl+V	Seçimi kopyala, kes, yapıştır.

VERİ

Adlandırılmış tablolar. Her tablo satır ve sütunlardan oluşur. Koddan örnek verecek olursak, `dget`, `dset`, `dstr`, `drows` ve `dcols` bu tabloları kullanır. Koddaki satır ve sütun numaraları 0'dan başlar.

Başlıkta yer alan sütun türleri:

- **bool** — bir bit.
- **2** — iki bit.
- **4** — dört bit.
- **8** — bir bayt.
- **16** — Kimlikler ve daha büyük sayılar için 16 bit.
- **spr** — bir sprite kimliği.
- **str** — sistem yazı tipiyle çizilmiş metin.
- **str8** ve **str10** — aynı metin, daha uzun çizilmiş hali.
- **f16** ve **f32** — yarı hassasiyetli ve tam hassasiyetli kayan noktalı imleçler.

İmleci sütun başlığında tutarken, Ctrl+kaydırma tuşu o sütunun türünü değiştirir. Düz kaydırma tuşu, AYARLAR → VERİ DÜZENLEYİCİ'nin seçtiği şekilde sütunlar arasında veya satırlar boyunca hareket eder. Ctrl tuşu bu yönü tersine çevirir; bu nedenle, fare tekerleği sütunlar moduna ayarlandığında, başlık üzerinde Ctrl+kaydırma tuşuyla tür değiştirilir.

Ok tuşları hareket ettirir. Enter tuşu hücreyi düzenler. AYARLAR → VERİ DÜZENLEYİCİ:

- **Tekerlek** — sütunları veya satırları kaydırın.
- **Renkler** — tablo, başlık ve seçim alanı.

Varlıklar

ASSETS, tüm projeyi değil, açık olan parçayı içe ve dışa aktarır. Sprite'lar, PAINT'te açık olan sprite'ı, açık olan sayfadaki sprite'ı kullanır. Harita, açık olan haritadır. SFX, açık olan desendir. Müzik, açık olan şarkıdır. Veri, açık olan tablodur. Kod ise istisnadır: her sekme yazılır ve her sekme geri okunur.

Tür	Dosya	Neler hareket eder?
Sprite'lar	`.png`	Açık sprite.
Harita	`.p37map`	Açık harita.
SFX	`.p37sfx`	Açık desen.
Müzik	`.p37mus`	Açılış şarkısı. Desen numaraları olduğu gibi kalır. Ses efektlerine ihtiyacınız varsa bunları ayrı olarak dışa aktarın.
Veri	`.p37dat`	Açık masa.
Kod	`.lua`	Her sekme.

Ayarlar

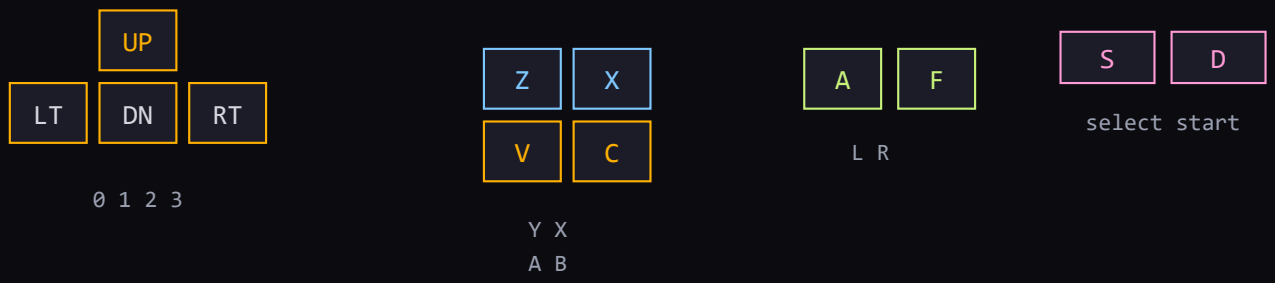
Kök liste:

Sayfa	Ne saklıyor?
PROFİLLER	Tüm ayarları kaydedin ve yeniden yükleyin. Yeni ayar ekleyin, yeniden adlandırın, üzerine yazın, silin. Profili metin olarak kopyalayın veya yapıştırın.
Menü	Pencere, tuşlar, menü renkleri, isteğe bağlı arka plan.
OYUN PANELİ	Önizleme ve oynatma penceresi.

KOD DÜZENLEYİCİ	Öne çıkan noktalar ve ipuçları.
BOYAMA EDİTÖRÜ	Fırça, palet, harita görünümü.
SFX EDİTÖRÜ	Sentezleyici ve nota girişi.
VERİ DÜZENLEYİCİ	Tekerlek yönü ve renkleri.
DİL	Menü ve ipucu dili.
LİSANSLAR	Lisans metinleri, Fusion Pixel yazı tipi ve SDL3 dahil.

KÜRESEL:

- **YERLEŞİM DÜZENİ** — panel düzeni, yukarıda.
- **SES SEVİYESİ** — ana seviye, 0 ila 10.
- **TUŞ ATAMALARI** — 8 oyuncunun her biri için tuşlar. Yakalamak için bir tuşa basın. Yön tuşları, ABXYLR, seç.
- **Sunum** — boş kareleri atla veya her kareyi çiz.
- **Tam ekran** — çerçevesiz.
- **Yenile** — pencere FPS. Sol ve sağ adımlar 60, 90, 120 ve maksimum. Enter tuşu 1000'e kadar bir sayı girer. 0, sınırlama olmadığını gösterir.
- **Vsync** — kareler arasında ekranın yenilenmesini bekler.
- **Ölçek** — tamsayı pencere çarpanı.
- **Gölgelendiriciler (Shaders)** — Oyunda değil, düzenleyicideki CRT ve filtreler.
- **Renk** — menü metni ve vurgulama.
- **Arka plan** — menünün arkasında yer alan, isteğe bağlı, 128 piksel boyutunda kırpılmış bir PNG dosyası. Temizle seçeneğiyle kaldırılır.
- **Editör imleci** — işaretçi şekli.
- **Fare** — hareketli işaretçi veya piksellere sabitlenmiş işaretçi.
- **Kullanıcı arayüzü hızı** — basılı tutulan bir komutun listede ne kadar hızlı tekrarlandığı.
- **Dosya Gezgini** — dosya listeleri bilgisayarın kök dizininden veya exe dosyasının bulunduğu klasörden başlar.



defaults — change them in SETTINGS, KEY BINDINGS, per player

Varsayılan klavye. A menülerde onaylamak içindir. B geri dönmek içindir. Bir oyunda bunlar 4 ve 5 numaralı tuşlardır. Seçme (S) 10 numaralı tuştur, menü kısayolu değildir.

Düğme	Rol	Varsayılan anahtar
0	Sol	Sol ok
1	Sağ	Sağ ok
2	Yukarı	Yukarı ok
3	Aşağı	Aşağı ok
4	A, onaylayın	V
5	B, arka	C
6	X	X
7	Y	Z
8	R	F
9	L	A
10	Seçme	S
11	Başlangıç	D

İnşa etmek

Bir platform seçin: Windows, Linux, Mac, Android, iOS veya Web.

- **Name** — derleme dosyasının adı.
- **Yazar** — Windows ve Linux'ta. Bu, kayıt dosyalarının klasör adıdır.
- **Steam uygulama kimliği** — Windows ve Linux'ta. Boş olması Steamworks'ün olmadığını gösterir. Doldurulmuş bir kimlik, `achieve(0)`...`achieve(15)`'yi Steam başarımları ACH_0...ACH_15 ile eşleştirir.

- **Id** — platformun istediği durumlarda kullanılan paket tanımlayıcısı.
- **Kodu gizle** — masaüstü ve mobil arayüzlerde. Oyun kartuşunun `peek` hızını yavaşlatır. Kilitleme özelliği değildir.
- **Tam ekran** — dışa aktarılan oynatıcı tam ekran modunda başlar.
- **Gölgelendiriciler** — oynatıcıya en fazla üç katman halinde işlenmiş gölgelendiriciler.
- **.p37 dosyasını indir** — Sadece web üzerinden erişilebilir. Duraklatma menüsünde indirme seçeneği gösterilir.
- **Simge** — bir sprite sayfasından bir karo: sayfa, boyut, x, y. Veya kırptığınız bir PNG kapak resmi.
- **Dışa Aktar** — klasörü seçin. Menüde "DERLEME KAYDEDİLDİ" veya "DERLEME BAŞARISIZ" yazısı görünür.

İnternet bağlantısı açıkken bir derleme yapın. Açılış ekranı, sunucuya anonim olarak bu derlemenin hash değerinin bilinip bilinmediğini sorabilir. Bu kontrol iki nedenden dolayı yapılır. Birincisi, PIX37'nin yazarı olan beni, başkasına vermemem gereken kopyalardan korur. İkincisi, oyunu yapan kişiyi korur. Başkasının oyununu alıp kendi oyununuzmuş gibi listelemek, Play (Android) veya iOS gibi telefon pazarlarında yaygın hale geliyor. Oyun yine de başlar. Makine çevrimdışıysa, giriş ekranında oyunun bağlı olmadığı belirtilir. Bu satır aynı zamanda oyuncuya kopyanın yasal bir kaynaktan gelmemiş olabileceğine dair bir ipucu verir. Bu küçük bir kontroldür, böylece oyununuz çalınıp izniniz olmadan başkası tarafından satılırsa, Play (Android), Steam veya itch.io gibi bir mağazada bu durumu savunabilirsiniz. Sunucu, hash değerinin ilk kaydedildiği zamanı hash değerinin yanına kaydeder ve daha sonra değiştirmez. Daha önceki zaman orijinal derlemedir. Bundan sonra kaydedilen hash değeri ise daha sonraki bir kopyadır.

Bu kontrol mekanizmasının oyun geliştiricilerini korumak için olduğunu düşünüyorum. Oyun motoru zaten onlar için işin büyük bir kısmını yapıyor ve bu da onlardan biri. Bunun tartışmalı olabileceğini anlıyorum. Eğer topluluk bu mikro-DRM'yi kaldırmanın daha iyi olduğuna karar verirse, ben de kaldıracağım.

Önceki bir editörden kalma eski bir oyun kartuşu, dil değişikliklerinden sonra başlatılamayabilir. Bu editörde `.pix37` projesini açın ve yeni bir `.p37` dosyası dışa aktarın.

PIX37.PARTY'ye bağlanın

PIX37.PARTY'ye bağlanma özelliği uygulama içerisinde yer almaktadır. Tarayıcıda giriş sayfası bulunmamaktadır. Steam oturum bileti veya itch.io ile, kumanda ve fare kullanarak giriş yaparsınız. Servis, 2 ila 16 karakter uzunluğunda, benzersiz bir görünen ad gösterir. Lisans anahtarı ve Steam kimliği diğer oyunculara gösterilmez.

Hizmet şartlarından sonra, ana menüde Favoriler, Katalog, Sıralama, Oyuncular ve profiliniz yer almaktadır.

Geliştirici lisansı editördür. Oyunu yaparsınız, yüklersiniz ve yıldız bırakırsınız. Oyuncu lisansı ise oyuncudur. Kalpler, koşu bandında oynayan kişilerden gelir.

- **Katalog** — oyunlara göz atın. Filtreler, tür (en fazla üç), ekran boyutu, oyun kartuşu boyutu ve yüklenen oyunun `.pix37` projesi içerip içermediğini kapsar.

- **Favoriler** — işaretlediğiniz oyunlar.
- **Yükleme** — Geliştirici lisansı gerektirir. `.p37` dosyasını gönderirsiniz ve `.pix37` projesini ekleyebilirsiniz. `.p37`, paketlenmiş bayt kodudur. Okunması zordur ve kasıtlı olarak gizlenebilir. `.pix37` ise aslında üzerinde çalıştığınız proje dosyasıdır: kaynak kod, resimler ve sesler, başka birinin açıp öğrenebileceği bir biçimde. Bunu göndermek, diğer geliştiricilerin sıkıştırılmış bir oyun kartuşu yerine oyunu incelemesine olanak tanır. Başlık, yazar, türler, kapak, simge ve başarı adları da bununla birlikte gönderilir. Gönderilen RAM sayısı, oyun kartuşunun ulaştığı en yüksek değerdir.
- **Yorumlar ve başarılar** oyun sayfasında yer almaktadır. Oturum açmışken bir başarı kilidini açmak, hem o hesapta hem de makinede kaydedilir.
- **Sıralama** — geliştiriciler, yıldızlara veya kalplere göre sıralanmıştır. Her satırda her iki sayım da gösterilir. Yazarın oyunlarını kalplere göre sıralanmış olarak görmek için bir isim açın.
- **Oyuncular** — hesaplar, kazanılan başarı sayısına göre sıralanmıştır.

Kod

Bu dil, Lua tarzı küçük bir betik dilidir. Bu bölüm, boş bir projeden başlayarak dili öğretir ve ardından her fonksiyonun kısa bir örneğini gösterir.

İlk program

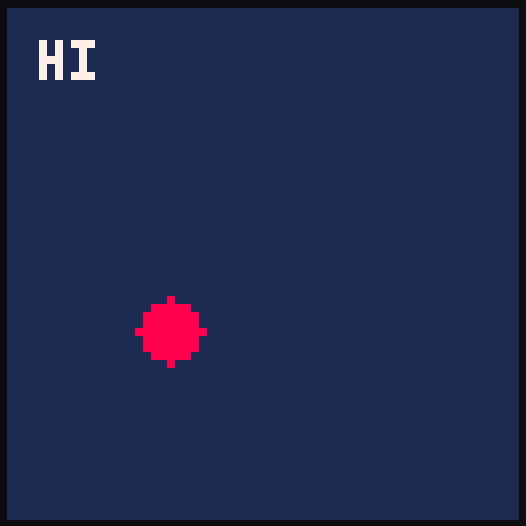
64x64 çözünürlükte, 4 bit renk derinliğinde bir proje oluşturun. CODE dosyasını açın. Sekmeyi aşağıdakiyle değiştirin:

```
_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 tuşu oyunu çalıştırır. Ok tuşları daireyi hareket ettirir. Escape tuşu oyunu durdurur ve düzenleyiciye geri döner.



Aynı üç komut: `cls(1)` lacivert ekranı boyar, `print` HI yazısını çizer ve `circfill` pembe daireyi çizer.

Neler çalışır ve ne zaman?

- `_init` bir kez çalışır.
- `_update` 60 Hz hızında çalışır. Hareket ve düğmeleri buraya yerleştirin.
- `_draw`, çizim işlemini gerçekleştirir. FPS argümanı olmadan, bilgisayar ve monitörün izin verdiği hızda çizim yapar. `_draw(30)` ise çizim hızını 30 FPS ile sınırlandırır ve mantıksal işlemleri 60 FPS'de bırakır.
- `_set_fps(n)` daha sonra bu sınırı değiştirir. 0 ise sınırı kaldırır.

İsimler büyük/küçük harf duyarlı değildir; düzenleyici bunları küçük harfle saklar. `function` ve `fun` aynı anahtar kelimedir.

Dil

Programlamanın temelleri

Bir program, oyun kartuşunun çalıştırdığı satırların bir listesidir. Her başlık önce satırın şeklini, ardından küçük bir örneğini gösterir. `--` ile başlayan bir satır bir yorumdur: sizin için bir not. Oyun bu satırı atlar.

Değişkenler

Değişken, bir değer tutan bir isimdir. İsmi siz seçersiniz. Sonraki satırlar bu ismi okuyabilir ve aynı isimde yeni bir değer saklayabilirler.

Şekil `name = value`'dir. `local name = value`, bu adı mevcut blok içinde tutar, böylece oyun kartuşunun geri kalanı onu göremez. `local` olmadan, ad globaldir ve her sekme onu görebilir.

```
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`, herhangi bir değerin olmadığını gösterir. `true` ve `false` ise sırasıyla evet ve hayır anlamına gelir. Daha önce hiç ayarlamadığınız bir isim `nil` olarak okunur.

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

Tablolar

Bir tablo, tek bir isim altında birçok değer barındırır. Bir liste sayılar kullanır. İlk öğe 1'dir, 0 değil. Bir kayıt isimler kullanır. Bir tablo başka bir tabloyu içerebilir.

Liste şekli: `{ value, value }`. Kayıt şekli: `{ name = value, name = value }`. Kayıt listesi, öğeleri tablolar olan bir tablodur.

```
-- 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
```

Durum ve vücut

Koşul, `hp > 0` veya `btnd(4)` gibi evet veya hayır cevabı olan bir sorudur. Gövde, bir bloğa girdiğinizde çalışan satır grubudur. `if` ve her döngü aynı yapıyı kullanır: bir koşul, ardından bir gövde, ardından `end`. `repeat`, `end` yerine `until` ile kapanır.

Eğer

`if`, gövdesini yalnızca koşul doğru olduğunda çalıştırır. `elseif` başka bir koşulu dener. `else` ise tüm koşullar yanlış olduğunda çalışır.

Şekil: `if condition then`, ardından gövde, sonra `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
```

Döngüler

Bir döngü, bir gövdeyi tekrarlar. `while` önce koşulu kontrol eder ve doğru kaldığı sürece devam eder. `repeat` gövdeyi çalıştırır, sonra kontrol eder, böylece gövde her zaman en az bir kez çalışır. `for` sayar. Son sayı dahil edilir. Üçüncü bir sayı olmadan adım 1'dir, bu nedenle başlangıç sonun ötesindeyse gövde çalışmaz. Geriye doğru saymak için negatif bir adıma ihtiyaç vardır.

```
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
```

bir masayı yürütmek

`all`, listenin ilk öğesinden başlayarak her bir öğesini ziyaret eder. `pairs` ise her bir adı ve o ad altında saklanan değeri ziyaret eder.

```
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
```

Fonksiyonlar

Bir fonksiyon, adlandırılmış bir tariftir. Yapısı `function name(arguments)`, ardından bir gövde, sonra da `end`'dir. Fonksiyonu, adını ve parantez içindeki ifadeyi yazarak çağırırsınız. `return` bir değer döndürür ve birden fazla değer de döndürebilir. Argümanlar, o fonksiyonun yerel değişkenleridir.

```
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`, etrafındaki yerel değişkenleri görebilen bir fonksiyondur. Bu bir closure'dır. Bir tablo bir closure içerebilir ve onu nokta ile çağırırsınız.

```
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
```

Evet, hayır ve seçim

`not` bir koşulu tersine çevirir. `and` yalnızca her iki taraf da doğru olduğunda doğrudur. `or`, taraflardan herhangi biri doğru olduğunda doğrudur. `a and b or c`, `a` doğru olduğunda `b`'yi, aksi takdirde `c`'yi seçer. `cond ? a : b` aynı fikri tek satırda ifade eder: koşul doğru olduğunda `a`, doğru olmadığında `b`.

```
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)
```

Matematik ve metin

`+ - * /` toplama, çıkarma, çarpma ve bölme işlemlerini ifade eder. `//` bölme işlemini yapar ve tam sayıyı korur. `%` kalanı ifade eder. `^` üs alma işlemini ifade eder. `..` iki metni birleştirir. `#t` listedeki öğe sayısını gösterir. Karşılaştırmalar `==` (aynı), `~=` veya `!=` (farklı) ve `< > <= >=`'dir. `x += 1` x'e 1 ekler. Aynı işlem `-= *= /= ^= %=` için de geçerlidir.

```
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
```

Yerli birinin yaşadığı yer

`local`, bloğunu kapatan `end`'ye kadar devam eder. `if` / `end` kendi başına bir bloktur. Bunu ayarlayan satırın üstünde yerel bir değişken belirtebilirsiniz, ancak bu satır çalışana kadar değer `nil`'dir.

```
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
```

Atlama

`::label::` bir satırı işaretler. `goto label` ise o satıra atlar, ancak yalnızca aynı fonksiyon içinde.

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

Kısaca hafıza

`peek` ve `poke`, 16 bitlik bir adres üzerinden canlı pikselleri inceler. Oyun kartuşunun ikinci bir kopyasını tutmazlar. `0x5F00` ile `0x5FFF` arasındaki adresler kayıtlardır: kamera, kırpma (sadece okunabilir), çizim rengi, dolgu deseni, üç palet yeniden eşlemesi ve sayfa numaraları. Bu deliğin her iki tarafındaki 65280 bayt, piksel penceresini oluşturur.

Ekran 65280 bayta sığdığında (64×64 ve 128×128 sığar, 256×256 sığmaz), ekran 0 adresinden satır satır başlar. Geriye kalan, `0x5F41` tarafından seçilen sprite sayfasıdır (0, sayfa 0'dır). Bu sayfa kalan alandan daha büyükse, `0x5F42` sayfayı seçer ve harita pencerede olmaz. Sayfa sığarsa, kuyruk, `0x5F43` tarafından seçilen haritadan bir veya iki baytlık harita hücreleridir ve `0x5F44`, kuyruğa sığmadığında haritayı sayfalara ayırır.

Ekran sığmadığında, tüm pencere 65280 pikselden oluşan tek bir sayfadır. `poke(0x5F40, n)` ekran sayfasını döndürür. n'nin 0'dan farklı olduğu `poke(0x5F41, n)`, bunun yerine n sayfasını gösterir ve bu modda 0 sayfasının görünümü yoktur. n'nin 0'dan farklı olduğu `poke(0x5F43, n)`, n haritasını gösterir ve bu, sayfaya göre önceliklidir. Bu modda 0 haritası yalnızca `mget` ve `mset`'dir.

Piksel veya kayıt defterine denk gelmeyen bir `peek` komutu 0 döndürür. Oradaki bir `poke` komutu hiçbir şey yapmaz. Sprite bayrakları, SFX ve DATA tabloları bu pencerede yer almaz. `fget`, `sfxget` ve `dget` komutlarını kullanın.

Pratik kullanım

Bu sayfanın alt kısmında her fonksiyon listelenmiş olup, PIX37 fonksiyonlarının pratik kullanım alanları da belirtilmiştir. Her örneğin altındaki bağlantılar, fonksiyonlar hakkında daha fazla bilgi sunmaktadır.



Pürüzsüz hareket

128×128 boyutlarında bir oyun kartuşu, yukarıdan görünümü. 1 numaralı karakter zaten çizilmiş. Oklar her yöne doğru hızlanıyor ve sürtünme sizi yavaşlatıyor. Karakterin kenarları sarılıyor: alttan çıkan yarısı üstte çiziliyor ve diğer kenarlar için de aynı şey geçerli.

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



Sonraki hamlenin engellenip engellenmediğini kontrol etme

64x64 boyutlarında bir oyun kartuşu. Zemin ve sağdaki duvar zaten haritada, 3 numaralı sprite ile boyanmış ve 0 numaralı bayrak bu sprite üzerine yerleştirilmiş. Oklar 1 numaralı sprite'ı hareket ettiriyor. Üstteki kelime, bir sonraki adımın düz bir karo ile çakışacağı zaman "engellendi", çakışmayacağı zaman ise "serbest" yazıyor.

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



Çevirmeli silme

A tuşuna basın. Silme efekti, _draw() fonksiyonu içinde değil, flip() fonksiyonu tarafından tek tek siyah satırlar halinde çizilir. Bu işlem sırasında _draw fonksiyonu devre dışı kalır, böylece her satır ekranda kalır.

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



Fonksiyonlar tablosu (Sınıf)

Sprite 1 zaten çizildi. `make_monster` bir sınıf gibi çalışır. Bir tablo döndürür ve `update` ve `draw` yöntemleri o canavarın kendi x ve y koordinatlarını tutar. En üstteki sayı, canavarın ne kadar yürüdüğünü gösterir.

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



Bir düğmeyi okumanın dört yolu

A, 4 numaralı düğmedir. Basılı tutun, şarj artar. Bırakın, şarj sıfıra döner. Yukarı ve aşağı tuşları basılı tutulduğu sürece tekrarlanır, böylece menü satırı hareket etmeye devam eder. btnd, basmanın ilk karesidir ve bu oyun kartuşu bunu yalnızca basılı tutmanın başlangıcı olarak kullanır.

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



Fareyle ateş et

Sprite 1 zaten ortada çizilmiş durumda. Beyaz nokta imleçtir. Sol tıklama, pikseli ona doğru hareket ettirir. Köşede basılı olan tekerlek hızı değiştirir.

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

 Bir isim yazın ve onaylayın.

Alanı açmak için A tuşuna basın. Yukarı ve aşağı tuşları yanıp sönen bir harfi değiştirir. A yazar, X siler, Y boşluk bırakır ve B onaylar. Ardından ekranda adınız görünür.

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

 Oyuncunun üzerindeki kamera

128x128 boyutlarında bir oyun kartuşu. 1 numaralı karakter, karo döşeli bir zeminin ortasında başlıyor. Oklar hareket ediyor. Kamera, karakteri merkezde tutuyor. Köşedeki x ve y eksenleri kaydırılmıyor, çünkü hiçbir argüman alınmadan kullanılan camera() fonksiyonu, HUD için görüntüyü geriye doğru kaydırıyor.

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

 Bir görsel romanın birkaç satırı.

Diyalog satırları DATA formatında, İngilizce ve Lehçe olarak yazılmıştır. A tuşu devam etmeyi veya ekranda bir seçenek olduğunda kalmayı seçer. B tuşu bir seçenek olduğunda ayrılmayı seçer, aksi takdirde dili değiştirir. X tuşu bir seçenek gösterilirken dili değiştirir. Diyalog 8px yazı tipini kullandığı için Lehçe harfler çizilir.

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

 Peek and poke ile boyayın.

64x64 piksellik bir oyun kartuşu, yani 0 adresi sol üst pikseldir ve bir sonraki satır 64'ten başlar. Oyun kartuşu pikselleri yazar, bir satırı kopyalar, bir satırı temizler ve okuduğu değerleri yazdırır.

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

 Ekleyebileceğiniz bir liste

A, 8'e kadar bir sayı ekler. B, sondaki sayıyı çıkarır. X, ilk sayıyı çıkarır. Liste ekrana yazdırılır.

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

 Parçalarına ayrılmış bir isim

Oyun kartuşu ada,12,blue ifadelerini ayırır ve ismi, ilk harfini, 12'yi ve "blue" kelimesini yazdırır.

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

 Bir nota çalın, sonra değiştirin.

Desen 0 tek bir notadan oluşur ve her tekrar aynı ses yüksekliğindedir. A, o nota bittiğinde tekrar çalar. B, perdeyi yükseltir ve o notayı çalar.

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

Oyun kartuşuna yazılmış bir şarkı

0. satırda iki desen döngü halinde tekrarlanır, ilk iki kanalın her birinde birer tane. Satır 0 olarak kalır. ch0 ve ch1, her kanalın çaldığı notadır. Oyun kartuşundaki deseni, LEN'i ve LOOP'u düzenlemek, bu kontrollerin müzik panelinde ne işe yaradığını görmenin bir yoludur.

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

DATA'da üç canavar

VERİ panelinde zaten bir isim sütunu ve bir can puanı sütunu var; bunlarda sümük, yarasa ve göz bulunuyor. A, sümüğü yarasaya zarar veriyor. Sağdaki sayılar onun can puanını gösteriyor.

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

Skoru hatırlayın

A, bir puan ekler ve kaydeder. Oyundan çıkıp tekrar açarsanız, sayı hala oradadır. B ise bunu unutur.

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

Başka bir dosyadaki ikinci bölüm

İki oyun kartuşu aynı kayıt yuvası görevini paylaşıyor. Bölüm 1: A, oyunun bittiğini işaretliyor. Önce açılan Bölüm 2, Bölüm 1'i bitirmeyi söylüyor. Bundan sonra Bölüm 2 açılıyor ve B, kaydı siliyor, böylece Bölüm 1 tekrar oynanabiliyor.

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

Bir başarı

Sprite 8 zaten çizilmiş durumda. A tuşuna on kez basın. Bildirimde başarı adı belirtilir ve sprite 8 kilidi açıldığı için kalır. B tuşu, yerel dosyadaki bu başarıyı siler ve sayacı sıfırlar.

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

İçine girebileceğiniz bir oda

256x256 boyutlarında bir oyun kartuşu. Beyaz bir nokta imleçtir. Menüde sunucu, katıl veya sunucu listesi bulunur. Sunucu, "walk" adlı bir oda açar ve oyunu başlatır. Katıl, bir adres (rakamlar, nokta, iki nokta üst üste)

kaydırır ve B bağlanır. Sunucu listesi yukarı ve aşağı hareket eder ve A satıra katılır. Oyunda oklar 1 numaralı sprite'ı hareket ettirir. Bir jeton paylaşılır: sunucu onu hareket ettirir ve onu almak o oyuncunun puanına 1 ekler ve bir sonrakini oluşturur. Sizinki de dahil olmak üzere her sprite'ın yanında o puan gösterilir. Köşede kaç oyuncunun olduğu, LAN adresi ve sunucunun genel adresi gösterilir. B ayrılır: siz ve diğerleri menüye dönersiniz ve bir sonraki sunucu 0 puanla yeni bir oda başlatır.

`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`

 Bekleyen diyalog

A, bir sonraki satırı gösterir. Fonksiyon, A'ya tekrar basana kadar her yield ifadesinde duraklatılır.

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

 Yarım saniye bekleyin

Kelime yaklaşık yarım saniye sonra "..."dan "merhaba"ya dönüşüyor. Eş zamanlı işlem duraklatıldığında ekran çizim yapmaya devam ediyor.

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

 Birer birer gelen daireler

Beş daire yukarıdan, iki kare arayla damlayarak aşağı düşüyor.

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

 Saatten iki kare

Sprite 1 ve sprite 2, aynı karakterin iki farklı karesidir. Resim saniyede dört kez değişir.

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

Her fonksiyon

Köşeli parantez içindeki argümanlar isteğe bağlıdır. Tablo ve metin dizilerinin indeksleri 1'den başlar, ancak DATA satır ve sütunları 0'dan başlar. Renkler palet indeksleridir.

İndeks

`_çizim`
`_init`
`_mp`
`_mp_fail`
`_mp_join`
`_mp_leave`
`_mp_update`
`_set_fps`
`_güncelleme`

karın kasları
açık
achdef
başarmak
elde edildi
ekle
hepsi
ileri sürmek
atan2
bant
değil
Bor
çok
ve benzeri
btnp
bu kadar
bxor
kamera
tavan
saat
daire
daire dolgusu
tıkla
tıkla
tıklama
klip
cls
birlikte yaratmak
renk
çekirdek
çünkü
kostüm
saymak
imleç
Dcol'ler
del
Şarküteri
kaydet
rakam
drowlar
dset
dstr
siktir et
doldurmak
çevir
flört
her biri
fset
bilgi
çiftler
kapatma
anahtar
anahtar durum
anahtar dizisi
çizgi
yük
günlük
işr
harita
maksimum
memcpy
memset
Mget
orta
dk.
fare
mset
müzik
müzikstat
duyurmak
ara
sonraki
ana bilgisayar
katılmak
nkb
nkbstr
nlan
ayrılmak
liste
dinle
not
n oyuncu
gönder
sersemletme
ordu
oval
oval dolgu
paket
çiftler
dostum
palrgb
palto
PC araması
göz atmak
göz2
göz4
mükemmel
sayfa bilgisi
dürtmek
dürtmek2
dürtmek4
baskı
baskı10
baskı8
pset
doğrultu
düzeltme
rnd
çürük
çürük
doğru
yeniden doldur
kaydet
kayıt yuvası
ekran
kaydırma
seç
ayar haritası
set paleti
setspr
setteks
gerileme
sfx

sfxget	günah	istatistik
sfxhdr	şarkı sözü	alt
sfxset	şarkı seti	zaman
sfxstat	bölünmüş	çizgi
sget	ilkbahar	ton
işaret	kare	tostr
sessiz	küçük parça	paketi açmak
gösteri istatistikleri	set	vay be
shr	Sspr	verim

Yaşam döngüsü

_init

`_init() ... end`

Oyun kartuşu başlatıldığında bir kez çalışır. Herhangi bir hook'un dışında kalan kod da _init'e dahil edilir.

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

_update

`_update() ... end`

Oyun mantığı. _draw sınırlandırılmış olsa veya sunucu daha yavaş olsa bile, 60 Hz'de çağrılır. Düğmeleri buradan okuyun.

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

_draw

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

Tek bir kare çizer. Hiçbir argüman verilmezse veya 0 değeri girilirse, sunucunun sunduğu sıklıkta çalışır. Saniyede n kare ile sınırlamak için n (1..999) değerini geçirin. _update 60 Hz'de kalır.

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

_set_fps

`_set_fps(n)`

Oyun çalışırken çizim sınırını değiştirir. 0 sınırsız anlamına gelir. n, 0 ile 999 arasında sınırlandırılmıştır.

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

flip

`flip()`

Ekranı gösterir ve bir sonraki kareye kadar bekler. Yalnızca kendi kare döngünüzü çiziyorsanız kullanın. Bundan sonra meşgul beklemesi yapmayın.

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

time

`time()`

_init'ten bu yana geçen saniye sayısı.

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

Giriş

btn

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

1, b düğmesi basılı tutulduğu sürece, aksi halde 0. p oyuncuyu temsil eder, 0..7. Argüman verilmediğinde 0 numaralı oyuncunun bit maskesini döndürür. Düğmeler: 0 sol, 1 sağ, 2 yukarı, 3 aşağı, 4 A, 5 B, 6 X, 7 Y, 8 R, 9 L, 10 seç, 11 başlat.

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

btnd

`btnd(b, [p])`

Yalnızca b düğmesine basıldığında _update çerçevesinde doğrudur.

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

btnp

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

Basıldığında doğru, basılı tutulduğunda tekrar doğru: 30 tik gecikme, ardından her 8 tikte bir. Hiçbir düğmeye basılmadığında maske döndürür.

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

btnu

`btnu(b, [p])`

Yalnızca düğme bırakıldığı anda doğrudur.

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

mouse

`mouse()`

Farenin x, y koordinatlarını oyun içi piksel cinsinden döndürür.

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

clickl

`clickl([p])`

Sol fare düğmesi. clickl() basılı tutulur. clickl(1) basma kenarıdır.

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

clickr

`clickr([p])`

Sağ fare düğmesi. clickl ile aynı 1 argümanlı kenar.

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

clickm

`clickm([p])`

Orta düğme veya fare tekerleğinin bir tıklaması.

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

scroll

`scroll()`

Bu karede fare tekerleğinin hareketi gösterilmiştir. Pozitif ve negatif hareket, tekerleğin yönüne bağlıdır.

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

keyon

`keyon()`

Düğmelerle kontrol edilen bir metin alanı açar. Yukarı ve aşağı tuşları A-Z harflerini değiştirir. A tuşu o harfi yazar. X tuşu siler. Y tuşu boşluktur. B tuşu onaylar ve alanı kapatır. Klavye bu alana yazmaz.

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

keyoff

`keyoff()`

Metin alanını gizler.

```
keyoff()
```

keystr

`keystr([s])`

Alandaki metin, ardından çerçevedeki 1 (B) onaylıyor, sonra yukarı ve aşağı harfler değişiyor. Alan kapandığında o harf boş kalıyor. keystr("") metni temizliyor.

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

keystat

`keystat()`

Metin alanı açık olduğu sürece 1, aksi halde 0.

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

Çizim

screen

`screen()`

Oyun kartuşunun mevcut ekran boyutu piksel cinsinden. Önce genişliği, sonra yüksekliği döndürür. Bu, oyun çözünürlüğüdür, düzenleyici penceresinin çözünürlüğü değildir.

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

cls

`cls([col])`

Ekranı temizler. Varsayılan renk 0'dır.

```
cls(1)
```

color

`color(col)`

Sonraki bir çağrıda `col` parametresi atlandığında kullanılacak çizim rengini ayarlar. Argüman verilmediğinde geçerli rengi döndürür.

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

camera

`camera([x, y])`

Her çizim çağrısında ofsetler ayarlanır. camera() fonksiyonu 0, 0 değerlerine sıfırlanır.

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

clip

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

Çizimi bir dikdörtgenle sınırlandırır. Sıfırdan farklı beşinci argüman, mevcut kırpma alanıyla kesişir. clip() tam ekranı geri yükler. Kırpma kayıtlarına göz atma işlemi yalnızca okunabilir.

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

cursor

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

Koordinatsız bir sonraki yazdırma işleminin nereye yapılacağını ve isteğe bağlı olarak rengini ayarlar. cursor() x, y değerlerini döndürür.

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

pset

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

Ekran piksellerinden birini ayarlar. Piksel istatistiğine dahil edilir.

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

pget

`pget(x, y)`

Ekrandaki x, y konumundaki pikselin rengi.

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

print

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

3x5 boyutlarında metin çizer (UTF-8, Fusion yazı tipinden kırpılmış). x ve y koordinatları belirtilmezse, imlecin bulunduğu yere yazdırır ve bir sonraki satıra geçer.

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

log

`log(...)`

Argümanları boşluklarla ayırarak bir metin arabelleğine ekler. pix37_run bu arabelleği standart çıktıya yazdırır. Çizim yapmaz.

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

print8

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

Baskıyla aynı, 8 piksel Fusion yazı tipi kullanılarak.

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

print10

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

Baskıyla aynı, 10 piksel Fusion yazı tipi kullanılarak.

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

line

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

x0,y0'dan x1,y1'e giden çizgi. İki argüman önceki bitiş noktasından devam eder. `line()` bu zinciri sıfırlar.

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

circ

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

Daire ana hatları. Yarıçap piksel cinsindendir.

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

circfill

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

Dolu daire.

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

oval

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

Sınır kutusunun içindeki elips ana hatları. Köşeler dahildir.

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

ovalfill

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

Sınır kutusu içinde doldurulmuş elips.

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

rect

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

Dikdörtgenin dış hatları. Köşeler birbirine dahildir.

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

rectfill

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

Doldurulmuş dikdörtgen.

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

rrect

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

Yuvarlak köşeli dikdörtgen çerçeve. x, y sol üst köşeyi temsil eder. w ve h piksel cinsinden boyutları belirtir; her ikisi de 0'dan büyük olmalıdır. r köşe yarıçapıdır ve varsayılan değeri 0'dır. Daha kısa kenarın yarısına kadar sınırlandırılmıştır.

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

rrectfill

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

Doldurulmuş yuvarlak köşeli dikdörtgen. rrct ile aynı argümanlar.

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

fillp

`fillp([p])`

Daha sonra doldurulacak şekiller için 16 bitlik dolgu deseni. 0 dolu anlamına gelir. fillp() mevcut deseni döndürür. Bitler 4×4 deseninde ilerler.

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

pal

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

c0 rengini c1 rengine yeniden eşler. Mod 0 çizim paleti, 1 görüntüleme paleti, 2 dolgu deseni paletidir. pal() sıfırlama yapar.

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

palrgb

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

Bir renk paleti girişini RGB'ye ayarlar, her kanal için 0..255 aralığını kullanır.

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

palt

`palt([col, t])`

Şeffaflık. palt(col) veya palt(col, true) rengi şeffaf yapar. palt(col, false) rengi opak yapar. palt() ise her rengi opak yapar.

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

setpalette

`setpalette(n)`

Palet bankasını (n) seçer. setpalette() geçerli bankayı döndürür.

```
setpalette(1)
```

tline

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

Ekran üzerinde bir çizgi çizerken, harita konumlarından (mx, my) harita piksellerini örnekler. mdx, mdy ile harita örneğini çizgi boyunca ilerletir.

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

Sprite'lar ve haritalar

spr

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

n numaralı sprite'ı x ve y konumlarında çizer. w ve h piksel cinsinden boyuttur, varsayılan değer 8'dir. fx ve fy sıfırdan farklı olduğunda yatay ve dikey olarak çevirme yapar. tex, sprite sayfasının sekmesidir. rot, dönüş cinsinden açıdır (0..1), sin ve cos ile aynı yöndedir. 0, sprite'ı döndürmez.

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

sspr

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

Sprite sayfasının dikdörtgen bir bölümünü ekrana kopyalar, dw x dh ölçeğinde ölçeklendirir. Varsayılan ayarlar kaynak boyutunu korur.

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

sget

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

Sprite sayfasındaki bir pikselin rengi.

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

sset

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

Sprite sayfasına tek bir piksel çizer.

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

fget

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

Sprite bayrakları. fget(n) bayrak baytını döndürür. fget(n, f), bayrak biti f (0..7) ayarlanmış olduğunda doğrudur.

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

fset

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

Sprite bayraklarını ayarlar. fset(n, v) tüm baytı yazar. fset(n, f, v) bir biti ayarlar.

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

setspr

```
setspr(n)
```

Etkin sprite sayfasını seçer. setspr() geçerli sayfayı döndürür. settex ile aynı bankadadır.

```
setspr(1)
```

settex

```
settex(n)
```

setspr'ın takma adı.

```
settex(0)
```

map

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

Haritayı çizer. Haritanın cx, cy hücresi sx, sy ekranına çizilir. cw, ch değerinin 0 olması tüm haritayı ifade eder. Katman kabul edilir ve göz ardı edilir. tex, döşeme seti sayfasıdır.

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

mget

`mget(cx, cy)`

Harita hücresi cx, cy'deki sprite kimliği.

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

mset

`mset(cx, cy, n)`

N numaralı sprite'ı cx, cy harita hücrelerine yazar.

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

setmap

`setmap(n)`

n numaralı haritayı seçer. setmap() geçerli haritayı döndürür. Harita 0 gerçek bir haritadır.

```
setmap(1)
```

Hafıza

peek

`peek(addr)`

64 KB'lık pencerenin bir baytını okur. Pencere, canlı ekran, ardından sprite pikselleri ve ekran 65280 bayta sığdığında harita hücrelerinden oluşur. Daha büyük ekranlar poke(0x5F40, page) ile sayfalanır. Eşlenmemiş adresler 0 döndürür. Ayrıca @addr olarak da yazılır.

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

poke

`poke(addr, ...)`

Belirtilen adrese bayt yazar. Ek argümanlar sonraki baytları yazar. Eşlenmemiş adresler yok sayılır. Ayrıca $addr, val yazılır. Ekrana dokunmak piksel yazma olarak sayılır.

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

peek2

`peek2(addr)`

İşaretsiz 16 bitlik değer, little-endian, adreste.

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

poke2

`poke2(addr, val)`

İşaretsiz 16 bitlik bir değeri little-endian formatında yazar.

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

peek4

`peek4(addr)`

İşaretsiz 32 bit değer, little-endian.

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

poke4

`poke4(addr, val)`

İşaretsiz 32 bitlik bir değeri little-endian formatında yazar.

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

memcpy

`memcpy(dst, src, len)`

Aynı 64 KB'lık pencere içinde len bayt kopyalar.

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

memset

`memset(dest, val, len)`

dest adresinden başlayarak len baytlık alanı val değeriyle doldurur.

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

Matematik

abs

`abs(x)`

Mutlak değer.

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

flr

`flr(x)`

Eksi sonsuza doğru yuvarlanır.

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

ceil

`ceil(x)`

Artı sonsuza doğru yuvarlanır.

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

max

`max(a, b)`

İki değerden daha büyük olanı.

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

min

`min(a, b)`

İki değerden küçüğü.

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

mid

`mid(a, b, c)`

Üç değerin ortadaki değeri. Sınırlayıcı olarak kullanışlı.

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

sgn

`sgn(x)`

x'in işareti: -1, 0 veya 1.

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

sqrt

`sqrt(x)`

Karekök. Negatif bir argüman sayı değildir.

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

sin

`sin(x)`

Sinüs fonksiyonu x, sırayla 0, 0.25, 0.5, 0.75 değerleriyle çember etrafında bir tur atılarak kullanılır. Radyan cinsinden değil.

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

cos

`cos(x)`

Kosinüs de, sinüs ile aynı şekilde ele alınır.

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

atan2

`atan2(dx, dy)`

dx ve dy vektörlerinin 0 ile 1 arasında dönüş açıları.

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

rnd

`rnd([x])`

[0, x) aralığında rastgele sayı. Varsayılan x değeri 1'dir, bu nedenle rnd() 0..1 aralığındadır. Bir zar için, flr(rnd(6))+1.

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

srand

`srand(seed)`

Rastgele sayı üretecisini tohumlar, böylece daha sonraki bir rnd dizisi tekrarlanabilir.

```
srand(1)
```

band

`band(a, b)`

Bit düzeyinde VE işlemi. Ayrıca a & b olarak da yazılır.

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

bor

`bor(a, b)`

Bit düzeyinde VEYA. Ayrıca a | b şeklinde de yazılır.

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

bxor

`bxor(a, b)`

Bit düzeyinde XOR. Ayrıca ^^ b olarak da yazılır.

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

bnot

`bnot(a)`

Bit düzeyinde DEĞİL. Ayrıca ~a olarak da yazılır.

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

shl

`shl(a, n)`

Sola kaydır. Ayrıca << n şeklinde de yazılır.

shr

`shr(a, n)`

Sağa doğru aritmetik kaydırma. Ayrıca >> n şeklinde de yazılır.

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

lshr

`lshr(a, n)`

Mantıksal sağa kaydırma. Ayrıca >>> n şeklinde de yazılır.

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

rotl

`rotl(a, n)`

Bitleri sola döndürün. Ayrıca <<> n şeklinde de yazılır.

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

rotr

`rotr(a, n)`

Bitleri sağa döndürün. Ayrıca >>< n şeklinde de yazılır.

Tablolar ve değerler

add

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

v değerini t tablosuna ekler. Eğer i değeri verilmişse, 1 tabanlı indekse ekler.

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

del

`del(t, v)`

v değerine eşit olan ilk girdiyi siler.

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

deli

`deli(t, [i])`

i indeksini kaldırır ve o değeri döndürür. i indeksi belirtilmediğinde ise son girişi kaldırır.

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

all

`all(t)`

Dizinin ilgili bölümü üzerinde sırayla yineleyici kullanılır. Delikler, 1 tabanlı bir dizinin bittiği gibi ilerlemeyi durdurur.

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

foreach

`foreach(t, fn)`

fn(v) fonksiyonunu dizinin her elemanı için çağırır.

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

pairs

`pairs(t)`

Adlandırılmış alanlar da dahil olmak üzere her anahtar ve değer üzerinde yineleyici.

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

ipairs

`ipairs(t)`

Dizinin 1 tabanlı indeksine sahip bölümü üzerinde yineleyici.

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

next

`next(t, [k])`

k'den sonraki anahtar ve değeri döndürür. next(t) baştan başlar. Çiftler tarafından kullanılır.

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

count

`count(t, [v])`

Dizinin uzunluğunu belirten kısım. Eğer v değeri verilmişse, v değerine eşit olan giriş sayısını sayar.

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

pack

`pack(...)`

Argümanları yeni bir tabloya yerleştirir.

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

unpack

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

t[i]'den t[j]'ye kadar olan değerleri ayrı ayrı döndürür. Varsayılan değerler dizinin tamamını kapsar.

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

select

`select(i, ...)`

`select("#", ...)` ifadesi, ardından gelen argüman sayısını döndürür. `select(i, ...)` ifadesi ise i'den itibaren argümanları döndürür.

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

assert

`assert(c, [msg])`

c değeri yanlış veya nil olduğunda hata vererek durur. msg, varsayılan metni değiştirir.

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

pcall

`pcall(f, ...)`

f fonksiyonunu çağırır. Başarılı olursa true değerini ve sonuçları döndürür. Hata verirse false değerini ve hata metnini döndürür.

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

Dizeler

chr

`chr(n)`

0..255 baytlık bir kod aralığındaki bir karakter.

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

ord

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

i'den j'ye kadar olan indekslerdeki bayt kodları. İndeksler 1'den başlar. Varsayılan olarak ilk karakter kullanılır.

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

sub

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

i'den j'ye kadar olan alt dize, 1'den başlayarak. Sondaki kısmı almak için j'yi atlayın.

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

split

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

s'yi bir tabloya böler. sep varsayılan olarak virgüldür. Sayısal bir sep, parça genişliğidir. conv, doğru olduğunda parçaları tonum ile dönüştürür.

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

tonum

`tonum(s, [base])`

Sayısal bir değeri ayrıştırır. Taban değeri 2 ile 36 arasında olabilir. Metin sayısal değilse nil döndürür.

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

tostr

`tostr(v, [digits])`

Bir değeri metne dönüştürür. digits parametresi sayılar için ondalık basamak sayısını ayarlar.

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

Ses

sfx

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

n numaralı deseni çalar ve kanalı döndürür. ch parametresini atlarsanız veya -1 değerini geçerseniz, ilk boş kanal seçilir. ch -2, n numaralı deseni her kanalda durdurur. n -1, bir kanalı veya ch parametresi atlandığında her kanalı durdurur. n -2, döngüsel bir desenin döngüye girmek yerine bitmesini sağlar. sfx() hiçbir argüman olmadan -1 döndürür.

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

sfxget

`sfxget(n, step)`

Tek bir desen adımının tablosunu döndürür: [1] perde, [2] dalga, [3] ses seviyesi, [4] efekt. Perde 0..63'tür (C-0'dan D#5'e). Dalga 0..15'tir. Ses seviyesi 0..7'dir ve 0 sessizdir. Efekt 0..7'dir.

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

sfxset

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

Bir kalıp adımı yazar. Altı argümanın tamamı gereklidir.

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

sfxhdr

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

Desen başlığı. sfxhdr(n) başlık tablosunu döndürür. sfxhdr(n, key, val) bir alanı ayarlar: speed, loop_in, loop_out, level, noise, crush, dtune, rverb, dampn, swing, steps. 0..10 arası sayılar aynı alanları bu sırayla seçer. speed, TIC'dir, adım başına vuruş sayısıdır. Daha yüksek değer daha yavaştır. 15, dakikada 120 vuruştur. 16 bit, 1..255'i kabul eder. Daha dar sentezleyiciler TIC listelerine kilitlenir.

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

sfxstat

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

Desen kimliği ch kanalında veya -1 kanalında çalıyor. Varsayılan kanal 0'dır.

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

note

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

Desen olmadan tek bir nota çalar. Perde 0..63, dalga 0..15, ses seviyesi 0..7, efekt 0..7. "Ticks", TIC ile aynı 120 Hz'lik aralıklarla ne kadar süreyle çalacağını gösterir. "Ticks" atlanırsa, varsayılan bir adım olan 15 saniye sürer. Efekt bu süre boyunca çalışır, ardından kanal sessizleşir.

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

music

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

n numaralı şarkıyı en baştan başlatır. music(-1) durdurur. Zaten çalmakta olan şarkının kimliğiyle music fonksiyonunu çağırmak, şarkının çalmaya devam etmesini sağlar. music() fonksiyonu çalmakta olan şarkının kimliğini veya -1 değerini döndürür. fade milisaniye cinsindendir: şarkı yavaşça başlar veya n -1 olduğunda yavaşça biter. Ayarlanan mask bitleri, şarkı için kullanılan kanalları gösterir. 0, sekiz kanalın tamamını ifade eder. Diğer kanallar ses efektleri için boş kalır.

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

musicstat

`musicstat()`

Çalınan şarkının tablosu. `song` ve `row`, şarkı kimliği ve satır numarasıdır veya hiçbir şey çalınmadığında -1'dir. `step` ve `pos` sekiz rakamdan oluşur. `step[1]`, 0. kanalı, `step[8]` ise 7. kanalı temsil eder. `step`, nota indeksidir veya o kanal sessiz olduğunda -1'dir. `pos`, o notanın 0 ile 1 arasında ne kadar ilerlediğini gösterir. `tick`, bu satırın kaç kez çalındığını gösterir. `played`, `music()` fonksiyonundan bu yana başlatılan satır sayısını gösterir (şu anda çalınan satır dahil).

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

songget

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

Desen kimliği bir şarkı hücresinde saklanıyor. ch değeri 0..7. 255 değeri hücrenin boş olduğunu gösteriyor. Bu canlı çağrı; kod ipucu metni hala eski bir argüman sırasını gösteriyor.

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

songset

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

Bir şarkı hücresine desen kimliği yazar. Hücreyi temizlemek için 255 kullanın.

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

stat

`stat(n)`

Yalnızca ses. 16..23, n-16 kanalındaki ses efekti kimliğidir. 24..31, n-24 kanalındaki müzik kalıbıdır. Diğer tüm n değerleri 0 döndürür. Kare zamanlaması ve RAM, istatistik değil, performans ve bilgi verileridir.

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

Veri tabloları

dget

`dget(t, row, col)`

DATA panelinden t veri tablosunun bir hücresini okur. Satır ve sütun 0'dan başlar.

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

dset

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

Hücreye veri yazar. Metin sütunu için bir dize kullanın.

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

dstr

`dstr(t, row, col)`

Hücre metin olarak. Sayılar dönüştürülür.

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

drows

`drows(t)`

t tablosunun kaç satırı var?

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

dcols

`dcols(t)`

T tablosunun kaç sütunu var?

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

Kaydet ve başarılar

save

`save(k, v)`

v değerini k anahtarı altında kayıt dosyasına kaydeder. v bir sayı, bir dize veya bunların bir tablosu olabilir. Dosya sıkıştırılır.

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

load

`load(k)`

k anahtarını okur. Anahtar bulunmadığında nil döndürür.

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

delsave

`delsave(k)`

Kaydedilen dosyadan "k" tuşunu siler.

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

saveslot

`saveslot([name])`

Yazarın klasöründeki kayıt dosyasını seçer. İsim belirtilmediğinde, seçilen dosya oyunun adıdır. Dosya zaten mevcutsa 1 değerini döndürür.

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

achieve

`achieve(i)`

0 ile 15 arasında i başarımının kilidini açar. Yerel olarak saklanır. Bir Steam sürümü de ACH_i'yi ayarlar. Bir katalog oturumu bunu oyuncunun hesabına bildirir.

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

achieved

`achieved(i)`

1. Başarı i zaten kilidi açılmışsa.

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

achdef

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

i başarısı için ekranda 15 saniye boyunca gösterilen bildirim metni. spr isteğe bağlı 8×8 boyutunda bir karo, tex ise sayfasını temsil eder.

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

achclear

`achclear(i)`

Yerel dosyadaki başarımları siler. Daha önce ayarlanmış olan bir Steam ACH_i başarımı ayarlanmış olarak kalır.

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

Ağ

whoami

`whoami()`

Oturum açıkken oyuncu kimliğiniz 0 ile 15 arasındadır. Çevrimdışı olduğunuzda ise -1'dir.

```
id = whoami()
```

nhost

`nhost()`

Bu makine ana bilgisayar ise 1, aksi halde 0.

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

nplayers

`nplayers()`

0 ile 15 arasında kaç canlı kimlik (ID) mevcut?

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

nlisten

`nlisten()`

UDP sunucusunu çalıştırmaya başlar ve LAN adresini ip:port şeklinde döndürür.

```
addr = nlisten()
```

njoin

`njoin(addr)`

IP adresine veya IP:port adresine bağlanır.

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

nleave

`nleave()`

Oturumdan ayrılır. Diğer kullanıcılar _mp_leave olayını 0 nedeni ile görürler.

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

nsend

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

Kayıplı bir mesaj gönderir. İsim, olayın adıdır. En fazla dört ek değer takip eder. p, 0 ile 15 arasında bir değerle o eşe fısıldar; herkese göndermek için p değerini atlayın.

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

ncall

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

nsend ile aynı şekle sahip, ancak mesaj onaylanana kadar tekrar deneniyor. Art arda üç başarısızlık oturumu sonlandırıyor.

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

nlan

`nlan()`

nlisten işleminden sonraki mevcut LAN IP adresi:port numarası.

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

nstun

`nstun()`

STUN'dan aldığınız genel IP adresiniz:portunuz. Başka bir ağdaki misafir, bu ağa katılmayı deneyebilir.

```
pub = nstun()
```

setturn

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

Doğrudan yol başarısız olduğunda TURN yedeklemesi kullanılır. Boş argümanlarla setturn() çalıştırıldığında devre dışı bırakılır. Adres host:port şeklindedir.

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

nlist

`nlist(game)`

Belirtilen oyun adına ait açık oturumların tablosu. Her girişte adres, oyuncu sayısı ve isim bilgisi yer almaktadır.

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

nannounce

`nannounce(game)`

Kalp atışı, bu sunucunun oyun için nlist'te görünmesini sağlar. Sunuculuk yaparken zamanlayıcıyla çağırın.

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

nkb

`nkb([on])`

Adres alanı. nkb(1) açar, nkb(0) gizler, nkb() açık olup olmadığını gösterir. Yukarı ve aşağı tuşları bir karakteri (0-9, nokta veya iki nokta üst üste) değiştirir. A yazar. X siler. Klavye bu alana yazmaz.

```
nkb(1)
```

nkbstr

`nkbstr([s])`

Adres alanındaki metin, ardından karakterlerin yukarı ve aşağı doğru değişmesiyle değişiyor. nkbstr("") metni temizliyor.

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

_mp

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

Bu fonksiyonu siz yazıyorsunuz. Bir mesaj geldiğinde çalışır. `from` gönderen kimliğidir. `name` ise olaydır. En fazla dört değer takip eder.

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

_mp_join

`_mp_join(p)`

Bu fonksiyonu siz yazıyorsunuz. Bir eş, p kimlik numarasını aldı. Bu fonksiyon sizin için de çalışıyor; whoami() bu kimlik numarasıdır.

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

`_mp_leave`

`_mp_leave(p, why)`

Bu fonksiyonu siz yazıyorsunuz. P kullanıcısı ayrıldı. 0 normal bir ayrılmayı, 1 ise bağlantı kaybını temsil eder.

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

`_mp_update`

`_mp_update()`

Bu fonksiyonu siz yazıyorsunuz. Her 15 güncelleme karesinde, yaklaşık 4 Hz'de bir çalışıyor. Pozisyonları göndermek için iyi bir yer.

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

`_mp_fail`

`_mp_fail()`

Bu fonksiyonu siz yazıyorsunuz. Oturum sona erdi: sunucu ortadan kayboldu veya ncall üç kez başarısız oldu.

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

Eş zamanlı işlemler

`cocreate`

`cocreate(f)`

f fonksiyonundan bir eşzamanlı işlem oluşturur. Eşzamanlı işlem devam edene kadar çalışmaz.

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

coresume

`coresume(c, ...)`

`c` eşzamanlı yordamını, bir sonuç verene veya bir değer döndürene kadar çalıştırır. Başarılı olursa: `true` ve döndürülen değerler. Hatalıysa: `false` ve hata mesajı.

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

costatus

`costatus(c)`

"Askıya alınmış", "çalışır durumda" veya "ölü".

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

yield

`yield(...)`

Coresume işlemini çağıran kişiye kontrolü geri verir. Değerler, bu coresume işleminin sonuçları haline gelir.

```
yield(1)
```

Ölçüm

info

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

Oturum aktifken _draw fonksiyonunda canlı HUD'ı çizer: fps (1 saniyelik ortalama), upd, drw, px, max, ram, peak, ramc ve net değerlerini gösterir. info() fonksiyonu yazdırma imlecini kullanır. info(col) yalnızca rengi ayarlar. info(x, y) veya info(x, y, col) ise konumunu belirler. Sayılar küçük ekranlara sığması için B, k veya m harflerini kullanır. Çağrının kendisi piksel cinsinden sayılmaz.

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

showstats

`showstats([on])`

Bilgi ekranıyla aynı HUD, _draw işleminden sonra 0, 0 konumunda otomatik olarak çizilir, kamera ve kırpma dikkate alınmaz. showstats(0) bunu gizler. showstats(1) veya showstats() bunu gösterir.

```
showstats(1)
```

perf

`perf(n)`

Tek bir HUD numarası, böylece kendiniz çizebilirsiniz. 0 fps, 1 upd, 2 drw, 3 px, 4 max (bu saniyedeki en ağır kare), 5 ram, 6 ramc (ram artı paketlenmiş oyun kartuşu), 7 oyun kartuşu çalışmaya başladığından beri en yüksek ram kullanımı.

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