PICO-8 v0.1.10c
http://www.pico-8.com

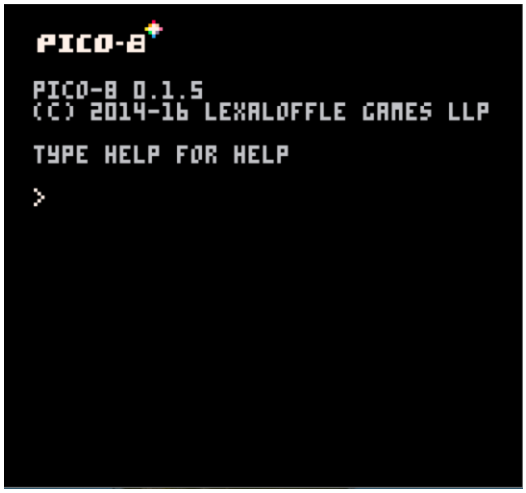
Autheur: Joseph White // hey@lexaloffle.com
PICO-8 est construit avec :
SDL2 http://www.libsdl.org
Lua 5.2 http://www.lua.org → voir license.txt
GIFLIB http://giflib.sourceforge.net
WiringPi http://wiringpi.com

Traduit de l'anglais par Jean-Marc QUERE, http://www.picoscope101.fr

Table des matières

Bienvenue à PICO-8!

PICO-8 est une console imaginaire pour faire, partager et exécuter des petits jeux et autres programmes. Quand vous démarrez le programme, la console vous accueille avec une ***Invite de commandes*** pour entrer directement des instructions Lua et fournit des outils simples pour créer des acteurs, une carte et des effets musicaux. Les limitations âpres de PICO-8 sont soigneusement choisies pour être amusantes à contourner, pour encourager la conception de petites réalisations originales et pour donner aux cartouches PICO-8 un caractère singulier.



Clavier

- Basculer en plein écran: [Alt]-[Entrée] ou [Command]-[F]
- Quitter: [Alt]-[F4] ou [Command]-[Q]
- Recharger/Lancer/Redémarrer une cartouche: [Ctrl]-[R]
- Sauvegarde rapide: [Ctrl]-[S]
- Pause : [P] ou [Echap.]
- Couper/Remettre le son : [Ctrl]-[M]
- Touches du joueur 1 : Touches fléchées Gauche, Droite, Haut, Bas + C/N + V/X
- Touches du joueur 2 : S, F, E, D + tabulation/Shift + A/Q

➔ D00.p8

Spécifications

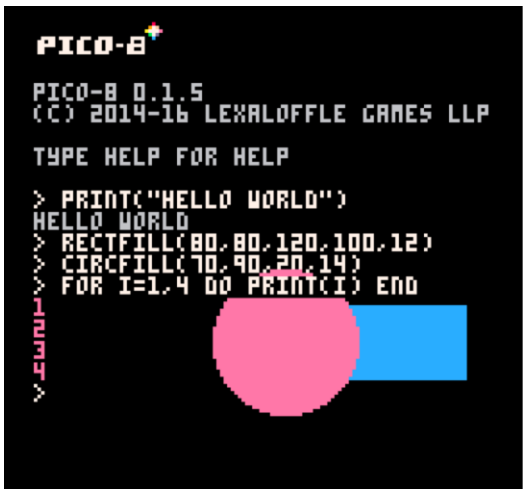
- Affichage : 128 x 128, palette de 16 couleurs fixes
- Entrées : 6 boutons x 2 joueurs
- Taille de la cartouche : 32kio
- Son : 4 canaux, 64 effets définissables
- Code : Lua (max. 8192 jetons de code source)
- Acteurs : une banque de 128 acteurs 8 x 8 (+ 128 partagés)
- Carte : une banque de 128 x 32 cellules (+ 128 x 32 partagées)

Hello World

Après l'amorçage de PICO-8, essayez de taper les commandes ci-après suivies d'un retour chariot (touche [Entrée]) :

```
PRINT("HELLO WORLD")
RECTFILL(80,80,120,100,12)
CIRCFILL(70,90,20,14)
FOR I=1,4 DO PRINT(I) END
```

> *PICO-8 affiche seulement des caractères en majuscule. Tapez normalement sans utiliser la touche de verrouillage des majuscules ([Caps Lock]) !*

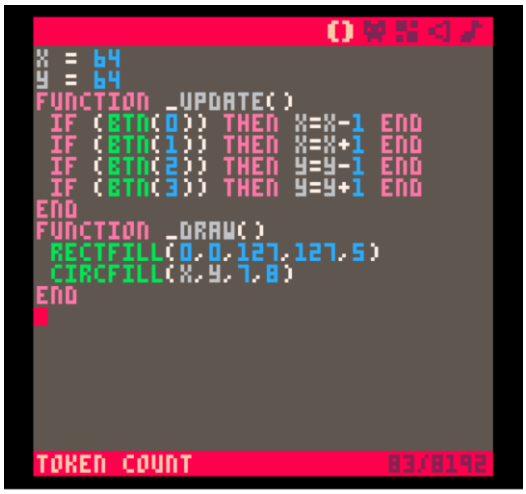


Vous pouvez construire un programme interactif en utilisant deux fonctions de rappel (callback) spéciales **`_update`** et **`_draw`** dans l'*Editeur de code*.

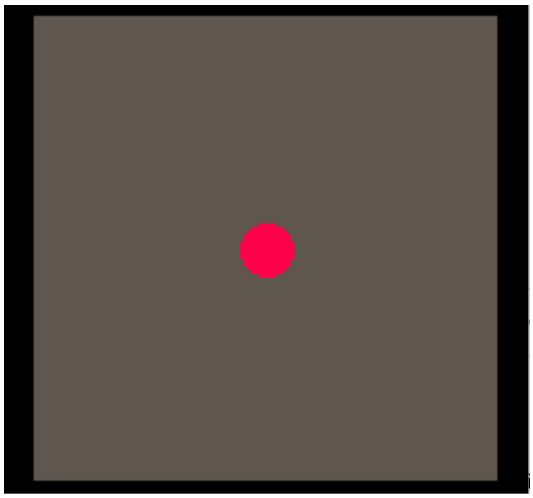
Exemple : le programme suivant vous permet de déplacer un cercle à l'écran avec les touches fléchées. Appuyez sur [Echap] pour accéder à l'*Editeur de code* et tapez ou copiez/collez le code suivant :

```
X = 64
Y = 64
FUNCTION _UPDATE()
 IF (BTN(0)) THEN X=X-1 END
 IF (BTN(1)) THEN X=X+1 END
 IF (BTN(2)) THEN Y=Y-1 END
 IF (BTN(3)) THEN Y=Y+1 END
END

FUNCTION _DRAW()
 RECTFILL(0,0,127,127,5)
 CIRCFILL(X,Y,7,8)
END
```



Pressez [Echap] pour retourner à la console et tapez **RUN** (puis [Entrée]) pour voir le programme en action.



Voir les cartouches d'exemple pour des programmes plus complexes.

Cartouches d'exemple

Ces cartouches sont incluses avec PICO-8 et peuvent être installées en tapant (à l'*Invite de commandes*) :

```
INSTALL_DEMOS
CD DEMOS
LS
```

API.....Présente la plupart des fonctions de PICO-8

JELPI.....Démo d'un jeu de plate-forme avec support de 2 joueurs

CAST.....Démonstration de raytracing (calcul d'une image de synthèse par projection de rayons lumineux, https://fr.wikipedia.org/wiki/Lancer_de_rayon)

MANDEL .....Explorateur Mandelbrot (fractale https://fr.wikipedia.org/wiki/Ensemble_de_Mandelbrot)

COLLIDE .....Exemple de gestion de collisions (acteur/mur)

BUTTERFLY .....Triangle de Serpinsky (fractale https://fr.wikipedia.org/wiki/Triangle_de_Sierpi%C5%84ski)

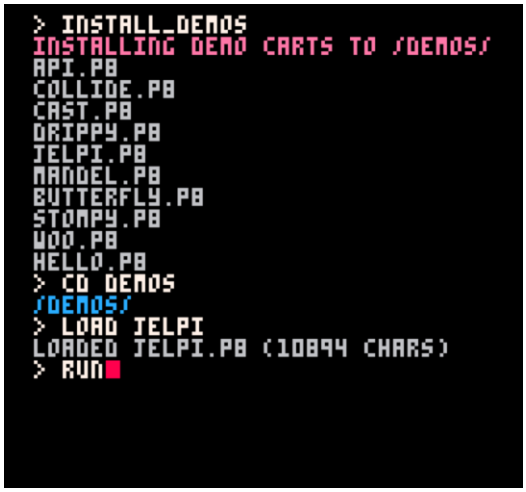
DRIPPY .....Dessiner un gribouillis dégoulinant

STOMPY.....Cartouche musicale

WOO.....Cartouche musicale

Pour lancer une cartouche (exemple : JELPI), tapez :

```
LOAD JELPI
RUN
```



Pressez [Echap] une première fois pour stopper le programme et de nouveau pour entrer dans l'*Editeur de code*.

Système de fichiers

Ces commandes peuvent être utilisées pour gérer les fichiers et les répertoires :

LS
DIR .....Liste le contenu du répertoire courant.

CD BWAH.....Change de répertoire (exemple : va dans le répertoire BWAH).

CD .. ..Remonte vers le répertoire parent.

CD /.....Retourne dans le répertoire racine (du le lecteur virtuel de PICO-8).

MKDIR BWAH .Crée un répertoire (exemple : BWAH).

FOLDER .....Ouvre le répertoire courant dans l'explorateur de fichier du système hôte.

LOAD BWAH ..Charge la cartouche BWAH à partir du répertoire courant dans la mémoire

SAVE BWAH....Sauve la cartouche en mémoire dans le répertoire courant (sous le nom BWAH)

Si vous souhaitez déplacer des fichiers, les dupliquer ou les supprimer, utilisez la commande **FOLDER** et effectuez l'ensemble de ces opérations à partir du système d'exploitation hôte.

L'emplacement par défaut du lecteur virtuel de PICO-8 est :

- pour Windows : C:/Users/Yourname/AppData/Roaming/pico-8/carts
- pour OSX : /Users/Yourname/Library/Application Support/pico-8/carts
- pour Linux : ~/.lexaloffle/pico-8/carts

Vous pouvez le modifier (ainsi que les autres réglages) en éditant le contenu du fichier pico-8/config.txt.

Astuce : si vous utilisez Dropbox, il est possible de définir le répertoire vers un dossier partagé Dropbox afin d'exploiter un même disque virtuel à partir de différents systèmes hôtes.

Sauvegardes

Si vous quittez sans enregistrer vos modifications ou écrasez un fichier existant une sauvegarde de la cartouche est automatiquement effectuée dans le répertoire pico-8/backup.

Configuration

config.txt

Vous pouvez trouver certains réglages dans config.txt. Editez le fichier lorsque PICO-8 ne fonctionne pas.

- Windows : C:/Users/Yourname/AppData/Roaming/pico-8/config.txt
- OSX : /Users/Yourname/Library/Application Support/pico-8/config.txt
- Linux : ~/.lexaloffle/pico-8/config.txt

Paramètres de la ligne de commande

pico-8 [*commutateurs*] [*nom-de-fichier*.p8]

-run *fichier* .....Exécute *fichier*(.p8.png, .p8) au démarrage

-width n .....Définit la largeur de la fenêtre ou de l'écran et ajuste l'échelle en correspondance (si non spécifiée)

-height n .....Définit la hauteur de la fenêtre ou de l'écran et ajuste l'échelle en correspondance (si non spécifié)

-windowed n .....Définit le mode d'affichage : 0 plein écran, 1 fenêtré

-sound n .....Volume sonore des effets 0..256

-music n .....Volume sonore de la musique 0..256

-joystick n .....Gestion par joystick à partir du joueur n (0..7)

-pixel_perfect .....1 pour conserver un ratio entier du nombre de pixels affichés pour chaque pixel de la console pico lors du redimensionnement de l'écran (par défaut)

-draw_rect *x,y,w,h* ...Position et taille de l'écran pico-8 dans la fenêtre (ou en plein écran)

-aspect n .....Aspect ratio 420 ⇔1:1, 560 ⇔4:3, 525 ⇔5:4, etc.

-splore .....Lance l'explorateur de cartouche suite au démarrage.

-home *chemin* .....Définit le chemin où stocker config.txt et les autres fichiers de données utilisateur

-desktop *chemin* .....Définit l'emplacement où les captures écran et gifs sont sauvegardés

-screenshot_scale *n* .Echelle des captures écran : x3 par défaut, soit 368x368 pixels

-gif_scale *n* .....Echelle des captures gif : x2 par défaut, soit 256x256 pixels

-gif_len n .....Définit la durée maximale du fichier gif en secondes (1..120)

-gui_theme *n* .....1 par active le mode « contraste élevé »

-timeout *n*.....Durée d'attente (s) avec échec lors d'un téléchargement de cartouche (30s par défaut)

Configuration du contrôleur (joystick/pad)

PICO-8 utilise le schéma de configuration des contrôleurs gérés par SDL2. Il détecte les contrôleurs usuels au démarrage et recherche le fichier de configuration utilisateur (sdl_controllers.txt dans le même répertoire que config.txt). sdl_controllers.txt comporte une association par ligne.

Pour générer un fichier de configuration utilisateur pour votre contrôleur, utilisez le programme controllermap fourni avec SDL2 ou essayez http://www.generalarcade.com/gamepadtool.

Partage de cartouches / Exportation HTML5

Vous disposez de 3 méthodes pour partager vos cartouches :

1. Partager le fichier .p8 ou .p8.png directement avec les autres utilisateurs de PICO-8
2. Poster la cartouche sur le site Lexaloffe pour obtenir une version jouable dans un navigateur Internet

   http://www.lexaloffle.com/pico-8.php?page=submit

---

*Voir **save()** pour la génération de la version **.p8.png**. (Chercher « **.png** »)*

---

3. Exporter la cartouche vers une page HTML5 autonome :

```
EXPORT FOO.HTML
```

Cette commande génère deux fichiers : foo.html and foo.js (Vous avez besoin des deux!)

Vous être libre de distribuer et d'utiliser les cartouches exportées comme vous l'entendez (à condition d'en être l'auteur ou d'avoir l'autorisation des auteurs).

Le fichier .html est conçu comme un modèle et peut être modifié selon vos besoins. Il comporte des boutons de réglages et un lien vers un site externe (site de partage des cartouches PICO-8 par défaut), ainsi qu'un script Javascript pour éviter le défilement de la page web lorsque l'internaute utilise les touches fléchées en jeu.

Veuillez noter que la largeur et la hauteur du lecteur peuvent être configurées, mais doivent normalement correspondre à la taille du *canvas* utilisé pour l'affichage (580x540 par défaut).

Jusqu'à 16 cartouches peuvent être empaquetées ensemble (cartouche actuelle et jusqu'à 15 autres à partir de leurs sauvegardes respectives au format .p8) :

```
EXPORT FOO.HTML DAT1.P8 DAT2.P8 GAME2.P8
```

Lors de l'exécution, les cartouches supplémentaires sont accessibles sous forme de fichiers locaux :

`RELOAD(0,0,0x2000,"DAT1.P8")` – charge la *Feuille des acteurs* à partir de DAT1.P8

`LOAD("GAME2.P8")` – charge et lance la cartouche GAME2

Seul le format de fichier .p8 est géré (n'oubliez pas de préciser l'extension).

Capture d'écrans, Vidéos et étiquettes de cartouches

Pendant que la cartouche fonctionne, utilisez :

- [F6] pour sauvegarder une capture écran sur le bureau
- [F7] pour capturer l'étiquette de la cartouche
- [F8] pour démarrer l'enregistrement vidéo
- [F9] pour sauvegarder la vidéo au format « .gif » sur le bureau (8 secondes max. par défaut)

Si [F6] à [F9] ne sont pas disponible sur votre système, utilisez [F1] à [F4].

L'étiquette de cartouche est enregistrée lors de l'emploi de **`save()`** avec le format « .p8.png ».

Vous pouvez enregistrer une vidéo à tout instant (l'enregistrement fonctionne en continu) : utilisez [F8] pour en définir le début si vous souhaitez disposer d'une vidéo d'une durée inférieure à 8 secondes. Pour changer la durée maximale du gif, affectez la valeur souhaitée (en secondes) au paramètre gif_len dans config.txt. Le format gif ne peut pas afficher 30 ips (images par seconde). PICO-8 utilise donc la fréquence disponible la plus proche disponible : 33.3 ips.

Résolution des problèmes

Points courants à vérifier :

- Les tableaux Lua commence à l'indice 1 et non 0. FOREACH débute à T[1] et non T[0].
- cos() and sin() prennent 0..1 au lieu de 0..>2*PI et sin() est inversé.
- sgn(0) renvoie 1.
- La moitié inférieure de la *Feuille des acteurs* et la moitié inférieure de la carte occupent le même espace mémoire.

Il est préférable d'utiliser seulement l'une ou l'autre si vous n'en maitrisez pas le fonctionnement.

- Les nombres PICO-8 ne vont que jusqu'à 32767.99.
- Bascule plein écran : utilisez la combinaison [Alt]-[Entrée] sur OSX ([Command]-[F] est utilisée pour la recherche).
- Pour exporter une cartouche .png, utilisez SAVE et pas EXPORT. EXPORT enregistre seulement la *Feuille des acteurs*.

Splore

SPLORE est un utilitaire pour parcourir et organiser les cartouches local et sur le site (lexaloffle.com). Tapez SPLORE puis la touche [Entrée] pour le lancer ou démarrez PICO avec l'option -splore. Il est possible de contrôler SPLORE intégralement avec un joystick :

- GAUCHE / DROITE pour choisir une liste de cartouche
- HAUT / BAS pour choisir une cartouche dans la liste affichée
- X, O ou MENU pour lancer la cartouche

Six rubriques présentent les cartouches disponibles sur le site Web : cartouches favorites, nouvelles cartouches, cartouches à succès, cartouches en cours d'élaboration, cartouches réalisées de façon collaborative et les cartouches associées à une compétition (JAM). Elles sont complétées par un moteur de recherche permettant de retrouver une cartouche d'après son nom ou celui de son auteur et un explorateur de fichiers (en local).

Si vous avez installé PICO-8 sur une machine sans accès internet, vous pouvez également utiliser INSTALL_GAMES pour ajouter une petite sélection de cartouches dans votre liste de cartouches préférées.

Editeur

Pressez [Echap] pour basculer entre l'*Invite de commandes* et l'*Editeur*.

Cliquez sur les onglets en haut à droite pour basculer d'un mode d'édition à l'autre ou pressez [Alt]-[Flèche Gauche] / [Flèche Droite]

ATTENTION : La seconde moitié de la ***Feuille des acteurs*** (banque 2 et 3) et la moitié inférieure de la carte partagent le même espace mémoire dans la cartouche. C'est à vous de décider comment les utiliser mais soyez averti que tout dessin dans la seconde moitié de la ***Feuille des acteurs*** altère les données dans la seconde moitié de la carte et vice versa.

Edition du code

- Maintenez [Shift] pour sélectionner (ou effectuez un cliquer-glisser à la souris)
- [Ctrl]-[X], [C], [V] pour couper, copier ou coller la sélection
- [Ctrl]-[Z], [Y] pour annuler et refaire
- [Ctrl]-[F], [G] pour chercher un texte et répéter la recherche
- [Ctrl]- [Flèche Haut] / [Flèche Bas] pour naviguer vers le début/la fin du programme
- [Alt]-[Flèche Haut] / [Flèche Bas] pour naviguer vers la fonction précédente/suivante
- [Ctrl]-[Flèche Gauche] / [Flèche Droite] pour passer d'un mot à l'autre
- [Ctrl][D] pour dupliquer la ligne courante
- [Tab] pour indenter la sélection ([Shift]-[Tab] pour désindenter)

Pour saisir les caractères spéciaux (glyphes) qui correspondent aux boutons, utilisez [Shift]-[L] / [R] / [U] / [D] / [O] [X] () ou pressez [Ctrl]-[K] pour basculer en mode glyphes.

En bas à droite de l'éditeur, vous pouvez voir le nombre de jetons utilisés. Un programme peut avoir au plus 8192 jetons. Chaque jeton est un mot (ex. nom de variable) ou un opérateur. Les paires de parenthèses et chaînes de caractères sont comptabilisées pour 1 jeton. Les virgules, points, points-virgules, mots clefs LOCAL et END ainsi que les commentaires ne sont pas comptés.

Edition des acteurs

L'éditeur d'acteurs est conçu pour être utilisé à la fois pour l'édition des acteurs et pour l'édition libre au niveau du pixel de la ***Feuille des acteurs***. L'explorateur en bas de l'écran fournit une vue 8 x 8 dans la ***Feuille des acteurs*** mais il est possible d'utiliser les outils de déplacement (main) et de sélection (cadre en pointillés) pour gérer des acteurs plus grands ou des surfaces quelconques.

Crayon

Effectuez un cliquer-déplacer sur l'acteur pour dessiner (s'applique à l'espace visible avec la couleur courante).

Maintenez la touche [Ctrl] en cliquant pour remplacer la couleur du pixel cliqué par la couleur sélectionnée dans la palette.

Utilisez le bouton gauche de la souris pour choisir une couleur (dans la palette).

Tampon

Cliquez pour appliquer le tampon.

Maintenez la touche [Ctrl] située à gauche du clavier en cliquant pour appliquer le tampon avec gestion de la transparence (la couleur de transparence, 0 par défaut, n'est pas copiée).

Cadre en pointillés

Outil de sélection. Raccourcis clavier : touche [Shift] située à gauche du clavier ou [S].

Pressez [Entrée] ou cliquez pour annuler la sélection.

En l'absence de sélection à l'échelle du pixel (dans la partie supérieure de l'écran) les opérations courantes sont appliquées à la sélection à l'échelle de l'acteur (dans la partie inférieure de l'écran).

Pour sélectionner des acteurs, [Shift]-déplacez la souris dans l'explorateur des acteurs.

Main

Outil de déplacement la zone éditée sur la ***Feuille des acteurs***. Raccourci clavier : [Espace].

Seau de peinture

Remplie avec la couleur courante (dernière sélectionnée dans la palette)

S'applique à la sélection courante, à défaut à la surface visible de la ***Feuille des acteurs.***

Touche supplémentaires

- [Ctrl]-[Z] pour annuler
- [Ctrl]-[C] pour copier la sélection (à l'échelle du pixel) ou les acteurs sélectionnés
- [Ctrl]-[V] pour coller à l'emplacement de l'acteur courant
- [Q], [W] pour passer à l'acteur précédent/suivant
- [1], [2] pour passer à la couleur précédente/suivante (dans la palette)
- [Tab] pour afficher la vue plein écran

La molette de la souris permet de zoomer (à partir du centre en plein écran)

Opération sur une sélection (de pixels ou d'acteurs) :

- [F] pour basculer horizontalement
- [V] pour basculer verticalement
- [R] pour tourner (la sélection doit être carrée)

Les touches fléchées déplacent la zone visible dans la direction de la flèche (en boucle dans le cas d'une sélection d'acteur).

Indicateurs d'un acteur

Les 8 cercles colorés sont des indicateurs pour l'acteur courant. Chacun peut être true (vrai, allumé) ou false (faux, éteint). Ils sont accessibles à l'aide des fonctions **`FSET`** et **`FGET`** et indexés à partir de 0 de la gauche vers la droite (0, 1, 2, … 7). Voir **fset()** pour davantage d'information.

Edition de la carte

La carte PICO-8 est un bloc de 128 x 32 (ou 128 x 64 en utilisant l'espace partagé avec la ***Feuille des acteurs***) de cellules 8 bits. Chaque cellule est représentée dans l'éditeur comme une référence à un acteur (0..255) mais vous pouvez utiliser cette espace comme vous le souhaitez.

Les outils sont similaires à ceux utilisés dans le mode d'édition des acteurs. Sélectionnez un acteur et cliquez puis déplacez la souris pour peindre les valeurs dans la carte.

Pour dessiner plusieurs acteurs à la fois, sélectionnez les acteurs dans l'explorateur en pressant [Shift] tout en déplaçant la souris.

Pour copier un bloc de cellules, utilisez l'outil de sélection et l'outil tampon pour coller.

Pour se déplacer dans la carte, utilisez l'outil de déplacement (main) ou maintenez [Espace]

[Q]/[W] pour passer à l'acteur précédent/suivant

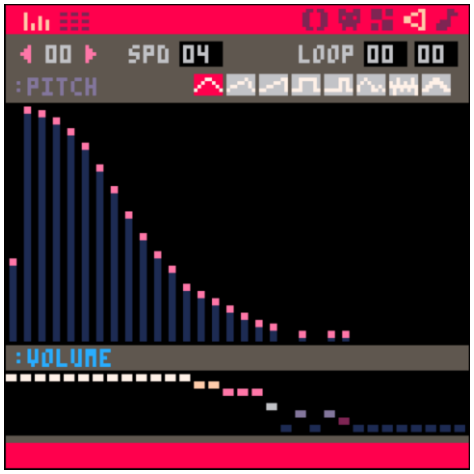
Editeur d'effets (sonore)

64 effets sonores sont disponibles pour les sons et la musique. Chaque effet a 32 notes. La note est définie par sa fréquence (C0..C5), son instrument (0..7), son volumen (0..7) et un effet (0..7). La valeur SPD indique la vitesse à la l'effet est joué en nombre de ticks par note : 1 rapide, 3 est 3x plus lent, etc. Les indications Loop start et Loop end précisent l'index de début et de fin de la boucle. Lorsque l'index de début (start) est supérieur ou égal à l'index de fin (end) la boucle est désactivée. Deux modes d'édition, respectivement orientés effet (Pitch) et musique (Tracker) peuvent être activés en utilisant les boutons en haut à gauche ou en alternance avec la touche [TAB].

Mode Pitch

Cliquez et déplacez la souris dans la zone de pitch pour définir la fréquence de chaque note en utilisant l'instrument courant (indiqué par la couleur).

- [8], [+] pour sélectionner l'effet édité (clavier principal FR, [-], [+] pavé numérique)
- [<], [>] change la vitesse de l'effet
- [Espace] pour jouer / stopper
- [Ctrl]-[A] pour sortir d'une boucle



Cliquez et déplacez l'instrument courant

[Shift]-cliquez un instrument pour remplacer toutes ses notes par l'instrument courant.

Maintenez [Ctrl] en traçant les fréquences pour accrocher le Do mineur (Cm) pentatonique.

[Cliquez avec le bouton Gauche] / [Cliquez avec bouton Droit] sur SPD pour diminuer / augmenter la vitesse (une valeur plus basse joue plus rapidement). Maintenez [Shift] pour appliquer un coefficient x4.

LP0 et LP1 déterminent les points de départ et d'arrivée de bouclage de l'effet.

- [Espace] pour jouer ou stopper
- [A] pour libérer un son qui boucle

Utilisez les boutons changement de mode de visualisation pour éditer les données de l'effet sous forme graphique ou dans le style tracker.

Mode Tracker

Chaque note montre : sa fréquence (en blanc), son octave (en gris), son instrument (en rose), son volume (en bleu) et son effet (en gris).

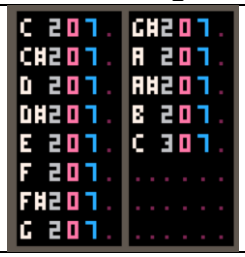
[Shift]-Cliquez un instrument, un effet, un volume pour l'affecter à toutes les notes.

Pour entrer une note, utilisez **a2z3er5t6y7ui wsxdcvgbhnj,** (disposition similaire à celle d'un piano). Les nouvelles notes sont affectées de l'instrument et de l'effet courant.

Pour supprimer une note, pressez [Retour-arrière] ou mettez le volume à 0.

Mettre le volume à 0 peut être utilisé pour supprimer plus facilement une succession de notes.

Effets : 0 aucun, 1 glissant, 2 vibrato, 3 drop, 4 progressivement plus fort, 5 progressivement moins fort, 6 arpeggio rapide, arpeggio lent. Les commandes Argeggio (6 rapide, 7 lent) agissent sur des groupes de 4 notes (à la vitesse 4, 8). Lorsque la vitesse de l'effet est inférieure ou égal à 8 les vitesses d'arpeggio sont divisées par 2.

a2z3er5t6y7ui	wsxdcvgbhnj,
	

Edition Musicale

[Cliquez avec le bouton Gauche] / [Cliquez avec le bouton Droit] sur le numéro de modèle pour indiquer quel effet sera joué pour ce modèle.

En haut à droite : utilisez les indicateurs **début de boucle**  , **fin de boucle**  et **stop**  pour contrôler le flux de lecture.

[Espace] pour jouer / stopper.

Syntaxe Lua

Les programmes PICO-8 sont écrits en utilisant la syntaxe Lua mais n'exploitent pas la bibliothèque standard de Lua. Veuillez trouver ci-après un bref résumé de l'essentiel de la syntaxe Lua. Pour plus de détails ou trouver des informations à propos de Lua, consultez www.lua.org.

Commentaires

`--` utilisez 2 traits d'union comme ça pour tout ignorer jusqu'à la fin de la ligne

Types et affectation

Les types en Lua sont les nombres, les chaînes de caractères (textes), les booléens (**TRUE** = vrai / **FALSE** = faux) et les tables.

```
NUM = 12/100
S = "CECI EST UN TEXTE"
B = FALSE
T = {1, 2, 3}
```

La fonction TYPE renvoie le type de la variable passée en argument, soit sur la base de l'exemple précédent :

```
PRINT(TYPE(NUM)) -- "NUMBER"
PRINT(TYPE(S))   -- "STRING"
PRINT(TYPE(B))   -- "BOOLEAN"
PRINT(TYPE(T))   -- "TABLE"
```

Les nombres en PICO-8 sont tous de type 16:16 à virgule fixe (16 bits pour la partie entière, 16 bits pour la partie fractionnaire). L'étendue de leur domaine de définition va de -32768.0 à 32767.99

La notation hexadécimale avec partie fractionnelle optionnelle peut être utilisée :

```
0x11 -- 17 (1 x 16 + 1)
0x11.4000 -- 17.25
```

Conditions

```
IF NOT B THEN
  PRINT("B EST FAUX")
ELSE
  PRINT("B N'EST PAS FAUX")
END

-- AVEC ELSEIF

IF X==0 THEN
  PRINT("X EST EGAL A 0.")
```

```
ELSEIF X<0 THEN
  PRINT("X EST NEGATIF.")
ELSEIF X>0 THEN
  PRINT("X EST POSITIF.")
ELSE
  PRINT("CETTE LIGNE N'EST JAMAIS EXECUTEE.")
END


IF (4 == 4) THEN PRINT("EGAL") END
IF (4 ~= 3) THEN PRINT("NON EGAL") END
IF (4 <= 4) THEN PRINT("PLUS PETIT QUE OU EGAL A") END
IF (4 > 3) THEN PRINT("PLUS GRAND QUE") END
```

Boucles

```
FOR X=1,5 DO
  PRINT(X)
END
-- affiche 1,2,3,4,5
X = 1
WHILE(X <= 5) DO
  PRINT(X)
  X = X + 1
END
-- affiche 1, 2, 3, 4, 5
FOR X=1,10,3 DO PRINT(X) END -- 1,4,7,10
FOR X=5,1,-2 DO PRINT(X) END -- 5,3,1
```

Fonctions et variables locales

```
Y=0
FUNCTION PLUS_UN(X)
  LOCAL Y = X+1
 RETURN Y
END
PRINT(PLUS_UN(2)) -- 3
PRINT(Y)          -- 0
```

Lua permet de vérifier une condition particulière et d'interrompre le programme avec un message d'erreur si celle-ci n'est pas remplie. Exemple : il est ainsi possible de vérifier que la valeur de X soit toujours un nombre positif inférieur à 100 dans la fonction PLUS_UN.

```
FUNCTION PLUS_UN(X)
  ASSERT(X>0 AND X<100,"X NE VERIFIE PAS 0<X<100 !")
  LOCAL Y = X+1
 RETURN Y
END
```

L'emploi d'ASSERT permet de vérifier les conditions requises avant d'effectuer un traitement et ainsi de détecter plus facilement les erreurs sans avoir à en rechercher l'origine.

Tables

En Lua, les tables sont des collections de paires (clef, valeur) où les types des clefs et des valeurs peuvent être mélangés. Elles peuvent être utilisées comme des tableaux indexés par des entiers.

```
A={} -- crée un table vide
A[1] = "BLAH"
A[2] = 42
A["FOO"] = {1,2,3}

-- Les tableaux utilisent une indexation débutant à 1 par défaut
A = {11,12,13,14}
PRINT(A[2]) - 12


-- La taille d'une table indexée par des entiers de façon
-- séquentielle à partir de 1 :
PRINT(#A)   -- 4


-- Les index de type chaîne de caractères peuvent bénéficier de la
-- notation pointée
JOUEUR = {}
JOUEUR.X = 2 - est équivalent à PLAYER["X"]
JOUEUR.Y = 3
```

```
-- voir aussi la section sur les tables dans le référentiel
-- des API ci-après.
```

Prise en main rapide de PICO-8

PICO-8 autorise également des formulations non standard plus concises.

1. IF THEN END sur une seule ligne peut être exprimée dans THEN ni END

```
IF (NOT B) I=1 J=2
-- est équivalent à : IF (NOT B) THEN I=1 J=2 END
-- important : la condition doit être placée entre parenthèses.
```

2. Opérateurs mathématiques unaires (à un seul opérande)

```
a += 2  -- équivaut à : a = a + 2
a -= 2  -- équivaut à : a = a - 2
a *= 2  -- équivaut à : a = a * 2
a /= 2  -- équivaut à : a = a / 2
a %= 2  -- équivaut à : a = a % 2
```

3. L'opérateur !=

Ce n'est pas un raccourci mais PICO-8 accepte **!=** en replacement de **~=** pour « non égal à »

API (Interface de Programmation d'Application)

PICO-8 est construit à partir de langage de script Lua mais n'inclut pas la bibliothèque standard Lua. A la place, il offre une Interface de Programmation d'Application (API) au design minimaliste (comme tout ce qui fait le charme de PICO-8). Pour une démonstration de la plupart des fonctions proposées, voir /DEMOS/API.P8

Les fonctions sont décrites ci-après sous la forme :

nom-de-fonction *paramètre* [*paramètre optionnel*]

L'emploi des fonctions système en ***Invite de commandes*** admet l'omission des parenthèses et des guillemets délimitant les chaînes de caractères.

```
LOAD BLAH.P8 --> LOAD("BLAH.P8")
```

Système

SPLORE
Interface de navigation pour accéder aux cartouches (présentes en local ou sur le site officiel : http://www.lexaloffle.com).

LOAD *nom-du-fichier*
SAVE *nom-du-fichier*
Charge ou enregistre une cartouche

Utilisez l'extension de fichier « .png » pour enregistrer au format image png (cartouche avec étiquette) – sinon le format texte est utilisé (.p8)

L'extension « .p8 » peut être omise dans tous les cas, elle est automatiquement ajoutée.

```
SAVE("BLAH") --> SAVE("BLAH.P8")
SAVE("BLAH.PNG") --> SAVE("BLAH.P8.PNG")
LOAD("BLAH.PNG") --> LOAD("BLAH.P8.PNG")
-- (si blah.png n'existe pas)
```

Le même principe s'applique au chargement. Si le ficher n'est pas trouvé, PICO-8 essaiera à nouveau en insérant « .p8 ». L'extension « .png » peut être omise (mais seulement lors d'un chargement).

```
LOAD ("BLAH") --> essaie "blah.p8.png" puis "blah.png"
-- (si "blah.p8" n'existe pas)
```

Une fois qu'une cartouche a été chargée ou enregistrée, le raccourci de sauvegarde rapide [Ctrl]-[S] peut être utilisé.

Utilisez le nom de fichier « @clip » pour charger à partir/sauvegarder vers le presse-papier.

Enregistrer des cartouches « .png » avec une étiquette (texte) et aperçu (image)

Pour générer l'image d'aperçu enregistrée avec la cartouche, lancez le programme d'abord, pressez [F7] pour capturer ce qui se trouve à l'écran. Les deux premières lignes du programme qui débutent avec « -- » sont apposées sur l'étiquette.

Exemple :

```
-- LES LEGENDES DES PLONGEURS
-- PAR LOOPY
```

Le format .png implique une restriction sur la taille du code. Lors de la sauvegarde au format .png, la taille du code compressé doit être inférieure à 15360 octets. Pour connaître la taille de votre code, utilisez la commande INFO. Cette limitation sur la taille du code ne s'applique pas au format .p8. Dans la majorité des cas, vous n'aurez pas à vous soucier de la taille du code compressé, la limite du nombre de jetons (8192) étant généralement atteinte en premier.

FOLDER
Ouvre le dossier des cartouches dans l'explorateur de fichiers du système hôte

LS
DIR
Liste tous les fichiers du dossier courant.

RUN
Lance l'exécution du programme (à partir du début).
Peut être appelé à l'intérieur du programme pour le réinitialiser

RESUME
Poursuit l'exécution du programme en l'état.

REBOOT
Redémarre la console.
Pratique pour commencer un nouveau projet.

STAT *x*

Renvoie une indication sur l'état du système selon la valeur de x :

0 .....Usage de la mémoire
1 .....Usage du processeur pour la dernière image générée (1.0 signifie 100%)
4.....Contenu du presse-papier (après emploi de [Ctrl]-[V]par l'usager)
16 à 19 .....Index des effets SFX en cours sur les canaux 0 à 3
20 à 23 .....Numéro des notes (0 à 31) sur les canaux 0 à 3

EXTCMD x

Commande système spéciale, fonctionne seulement lors de l'exécution d'une cartouche locale

"label" .....Etiquette de la cartouche
"screen" ...Enregistre une capture d'écran
"rec" .....Fixe le point de départ de la vidéo
"video" .....Enregistre une vidéo sous forme de .gif des *n* dernières secondes (8 par défaut)

TIME

Renvoie le nombre de secondes depuis le dernier reboot()

INFO

Affiche quelques informations sur la cartouche :
Taille du code, nombre de jetons (utilisés pour représenter le programme), taille compressée

FLIP

Bascule le tampon image d'arrière-plan à l'écran et attend la prochaine image (30 images par secondes)

Vous n'avez pas besoin de le faire _draw() l'appelle pour vous.

Si votre programme n'appelle pas flip() avant qu'une image ne soit prête et que la fonction _draw() n'est pas en cours, alors le contenu du tampon image est copié à l'écran.

PRINTH *str* [nom-du-*fichier*] [*écraser*]

Affiche une chaîne sur la console système du système hôte (utile pour la mise au point). Si le nom-du-fichier est indiqué, ajoute la chaîne str au fichier (dans le dossier courant, consultable avec la commande FOLDER). Lorsque la paramètre booléen écraser a TRUE pour valeur, la chaîne remplace le contenu actuel du fichier. Utilisez le nom de fichier « @clip » pour écrire dans le presse-papier. Vous pouvez utiliser STAT(4) pour accéder au contenu du presse-papier. Pour des raisons de sécurité, son contenu n'est accessible qu'après l'emploi de la séquence [Ctrl]-[V] par l'utilisateur durant l'exécution.

Structure d'un Programme

Il y a 3 fonctions spéciales qui, si elles sont définies par l'utilisateur, sont appelées pendant l'exécution du programme :

_UPDATE()
Appelée une fois par mise à jour de l'écran (soit 30 fois par seconde).

_DRAW()
Appelée une fois par image affichée.

_INIT()
Appelée une fois au démarrage du programme.

_draw() est normalement appelée 30 fois par seconde, mais si elle ne peut se terminer à temps, PICO-8 va tenter de l'exécuter 15 fois par secondes (et appeler _update 2 fois pour chaque image affichée pour compenser).

PICO-8 à 60 IPS

Si la fonction **_update60()** est définie à la place de **_update**(), PICO-8 fonctionne à 60 images par seconde. Les fonctions **_update60()** et **_draw()** sont appelées 60 fois par seconde. La moitié du processeur PICO-8 est disponible pour chaque image. Au-delà, la fréquence d'affichage est abaissée à 30 ips.

ATTENTION : toutes les versions de PICO-8 ne supportent pas l'affichage à 60 ips. Sur les machines qui ne le supportent pas la fonction **_update60()** est appelée 2 fois par image et **_draw()** à 30 ips. Vous pouvez vérifier le comportement de votre programme dans ces conditions à l'aide de l'astuce ci-après :

```
U60=_UPDATE60
_UPDATE60=NIL
FUNCTION _UPDATE() U60() U60() END
```

L'export HTML5 supporte les modes 30 et 60 images par secondes (depuis 0.1.9).

Graphisme

PICO-8 a une simple banque de 128 acteurs 8 x 8 plus une autre banque de 128 qui partagent le même espace que la moitié inférieure de la carte. L'ensemble de ces 256 acteurs est appelé la ***Feuille des acteurs*** et peut être envisagé comme une image de 128 x 128 pixels.

Toutes les opérations de dessin sont sujettes au contexte graphique courant. Ce qui inclut la position de la caméra (pour ajouter les offsets aux coordonnées), le mapping de la palette (pour recolorer les acteurs), le rectangle de clipping, et la couleur de tracé (qui à défaut d'être redéfinie par une fonction persiste).

Le contexte graphique est réinitialisé à chaque fois qu'un programme est exécuté. Ce qui est équivalent à l'appel de : clip(), camera(), pal(), color().

Index des couleurs

0 noir	1 bleu foncé	2 violet foncé	3 vert foncé
4 marron	5 gris foncé	6 gris clair	7 blanc
8 rouge	9 orange	10 jaune	11 vert
12 bleu	13 indigo	14 rose	15 pêche

CLIP ***[x y w h]***
Définit la région de clipping en pixels

CLIP() pour réinitialiser

PGET ***x y***
PSET ***x y [c]***
Lit ou fixe la couleur (c) d'un pixel en x, y.

SGET ***x y***
SSET ***x y [c]***
Lit ou fixe la couleur (c) d'un pixel de la ***Feuille des acteurs.***

FGET ***n [f]***
FSET ***n [f] v***
Lit ou fixe la valeur (v) d'un indicateur de acteur

f est l'index de l'indicateur 0..7

v est un booléen et peut être **TRUE** (vrai) ou **FALSE** (faux)

L'état initial des indicateurs peut être défini lors de l'édition des acteurs en intervenant sur la ligne des petits boutons colorés.

La signification des indicateurs est laissée à l'appréciation de l'usager. Ils peuvent être utilisés pour indiquer l'appartenant à groupe (« couche ») d'acteurs à dessiner sur la carte.

Si l'index n'est pas précisé, tous les indicateurs sont renvoyés / affectés sous forme d'un octet.

```
FSET(2, 1+2+8)   -- positionne les bits 0,1 and 3
FSET(2, 4, TRUE) -- positionne le bit 4
PRINT(FGET(2))   -- 27 (1+2+8+16)
```

PRINT ***str*** **[*x y* [*col*]]**
Affiche une chaîne de caractères.

Si seule la chaîne str est fournie, et que le curseur atteint la fin de l'écran, un retour-chariot (CR) et un saut de ligne (LF) sont automatiquement générés (comportement similaire à celui d'un terminal).

CURSOR ***x y***
Fixe la position du curseur et la marge du retour-chariot.

COLOR ***col***
Fixe la couleur par défaut utilisée par les fonctions de dessin.

CLS
Efface l'écran.

CAMERA [*x y*]
Fixe l'offset écran à –x, -y pour toute les opération de dessin.
CAMERA() pour réinitialiser

CIRC ***x y r*** **[*col*]**
CIRCFILL ***x y r*** **[*col*]**
Dessine un cercle ou un disque en x, y de rayon r.

LINE ***x0 y0 x1 y1*** **[*col*]**
Dessine une ligne de x0, y0 à x1, y1 avec la couleur col.

RECT ***x0 y0 x1 y1*** **[*col*]**
RECTFILL ***x0 y0 x1 y1*** **[*col*]**
Dessine un rectangle ou une surface rectangulaire.

PAL *c0 c1* [*p*]
Dessine toutes les instances de la couleur c0 en c1 dans les appels suivants.
pal() pour réinitialiser avec les valeurs par défaut (y compris la transparence)
Types de palette (p, 0 par défaut) :

- 0 palette de dessin : les couleurs sont mappées pendant lors du dessin (ex. pour recolorer des acteurs)
- 1 palette écran : les couleurs sont mappées à l'affichage (ex. pour les fondus)

c0 index de la couleur 0 à 15

c1 index de la couleur à mapper

PALT *c t*
Fixe la transparence pour la couleur indexée à t (booléen)

La transparence est utilisée par spr(), sspr() et map()

Ex. palt(8, true) – les pixels rouge ne sont plus dessinés

palt() réinitialise avec les valeurs par défaut : toutes les couleurs sont opaques sauf la couleur 0.

SPR *n x y* [*w h*] [*flip_x*] [*flip_y*]
Dessine le acteur n (0..255) à la position x, y

La largeur w et la hauteur h sont définies à 1, 1 par défaut et indiquent la taille (en acteurs) de la zone à dessiner

La couleur 0 est la couleur de transparence (non copiée) par défaut, voir palt()

flip_x = true applique un effet miroir horizontal

flip_y = true applique un effet miroir vertical

SSPR *sx sy sw sh dx dy* [*dw dh*] [*flip_x*] [*flip_y*]
Etire un rectangle à partir de la feuille de acteurs où (sx, sy, sw, sh) sont indiqués en pixels et pour le dessiner dans le rectangle (dx, dy, dw, dh) à l'écran

La couleur 0 est la couleur de transparence (non copiée) par défaut, voir palt().
dw, dh prennent pour valeurs sw, sh par défaut.
flip_x = true applique un effet miroir horizontal.
flip_y = true applique un effet miroir vertical.

IMPORT *nom-du-fichier.png*
EXPORT *nom-du-fichier.png*
Importe/Exporte la feuille des sprites à partir/à destination d'un fichier (image.png).

Très pratique pour récupérer des graphismes ou les travailler dans un logiciel externe.

Tables

ADD t v
Ajoute la valeur v à la fin de la table t.

Equivalant à t[#t+1] =v

```
FOO={}          -- crée une table vide
ADD(FOO, 11)
ADD(FOO, 22)
PRINT(FOO[2]) -- 22
```

DEL t v
Supprime la première occurrence de la valeur v dans la table t.
Les autres éléments sont décalés vers la gauche d'une position pour remplir les trous.
Important : v est la valeur de l'élément à supprimer, pas son index dans la table !
del() peut être appelé sur une table en cours de parcours

```
A={1,10,2,11,3,12}
FOR ITEM IN ALL(A) DO
IF (ITEM < 10) THEN DEL(A, ITEM) END
END
FOREACH(A, PRINT) -- 10,11,12
PRINT(A[3])       -- 12
```

ALL *t*
Utilisé dans les boucles FOR pour parcourir tous les éléments d'une table suivant l'ordre dans lequel ils ont été ajoutés.

```
T = {11,12,13}
ADD(T,14)
ADD(T,"HI")
FOR V IN ALL(T) DO PRINT(V) END -- 11 12 13 14 HI
PRINT(#T) -- 5
```

FOREACH *t f*

Pour chaque élément de la table t, appelle la fonction f avec l'élément comme seul paramètre.

```
FOREACH(T, PRINT)
```

PAIRS *t*

Utilisé dans les boucles FOR pour parcourir une table en fournissant à la fois la clef et la valeur de chaque élément. Contrairement à **all()**, **pair()** parcourt l'ensemble des éléments sans tenir compte de l'indexation. L'ordre n'est pas garanti.

```
T = {["HELLO"]=3, [10]="BLAH"}
T.BLUE = 5
FOR K,V IN PAIRS(T) DO
  PRINT("K: "..K.." V:"..V)
END
```

Affiche :

```
K: 10 v:BLAH
K: HELLO v:3
K: BLUE v:5
```

SETMETATABLE *t, mt*

Chaque table peut avoir une table associée appelée méta-table (mt). Les champs de la méta-table sont utilisés pour compléter ceux de la table. La clef __index indique la table dans laquelle la recherche se poursuit lorsqu'une clef souhaitée n'est pas trouvée dans la table (t). Dans ce contexte, les clefs d'une table :

- dotées d'une valeur s'appellent des propriétés (*t*["*nom*"]=*valeur*)
- associées à une fonction s'appellent des méthodes (FUNCTION *t*:*nom*(…) … END)

```
PIXEL={} -- crée une table vide nommée PIXEL
FUNCTION PIXEL:NEW(O) -- déclare la méthode NEW de la table PIXEL
  O=O OR {}
  O.X=O.X OR 0
  O.Y=O.Y OR 0
  O.C=O.C OR 7
  SETMETATABLE(O,SELF)
  SELF.__INDEX=SELF
  RETURN O
END
```

La table PIXEL comporte une clef « NEW » qui est associée à la fonction/méthode PIXEL:NEW. Elle accepte une table en paramètre : O. Si ce paramètre n'est pas précisé, O prend pour valeur nil

(⇔néant = état de l'inexistence). L'état booléen (logique) de nil (néant) est false (faux). Lorsque O n'est pas précisé lors de l'appel, la première ligne de la fonction « O=O OR {} » affecte une table vide ({}) à O. Le même principe est appliqué aux clefs X, Y et C pour affecter les valeurs par défaut : 0, 0 et 7. L'instruction SETMETATABLE et l'affectation à SELF.__INDEX associent la méta-table PIXEL à la table O (à ce stade O bénéficie de « tout le savoir-faire de PIXEL » dont la méthode DRAW).

```
FUNCTION PIXEL:DRAW() -- méthode DRAW (pour dessiner un PIXEL)
  COLOR(SELF.C)
  PRINT(SELF.X..",".SELF.Y.."=>"..SELF.C)
  PSET(SELF.X,SELF.Y)
END
```

Au sein d'une méthode (ex. PIXEL:NEW, PIXEL :DRAW), SELF est un référence à la table hôte de la méthode.

```
CLS()
CURSOR(0,64)
P0=PIXEL:NEW() -- P0 est un PIXEL (par défaut X=0,Y=0,C=7)
P1=PIXEL:NEW({X=10,Y=10,C=11}) -- P1 est un PIXEL (X=10,Y=10,C=11)
P0:DRAW() -- dessin P0
P1:DRAW() -- dessin P1
```

La création et l'affichage des pixels P0 et P1 illustrent le bénéfice de cette approche (objet). Lors de leur création, il est possible de préciser tout ou partie de leurs caractéristiques (propriétés) avec l'emploi d'une seule et même fonction (méthode). La méthode d'affichage (DRAW) n' pas besoin d'être déclarée pour chaque pixel : elle est mutualisée.

Pour faire le parallèle avec d'autres langages (C++, C#, JAVA,…) :

- La méthode NEW est assimilable à un constructeur.
- La méta-table PIXEL est assimilable à une classe.
- Les tables P0 et P1 sont assimilable à des objets (⇔instances de la classe PIXEL).

Les méthodes peuvent également adopter plusieurs comportements selon le type des arguments passés en paramètres (évaluable à l'aide de TYPE) ⇔ méthodes polymorphes.

Le principal intérêt de cette approche (programmation orientée objet) est de permettre la création de niveaux d'abstraction. Exemple : dissocier les conteneurs (listes chaînées, dictionnaire,…) des contenus (entiers, chaines, tables…) pour mutualiser des traitements inhérents à la vie du conteneur (création, ajout, suppression, recherche,…) quel qu'en soit le contenu. Le sujet est très vaste et conduit à la conception par patrons (design patterns).

---

*Si à ce stade vous vous sentez perdu, ce n'est pas grave : l'instruction **setmetatable** et l'orientation objet ne sont pas indispensables. De très nombreuses cartouches ont été réalisées sans celles-ci et de nombreuses autres continueront à l'être.*
Par contre, si l'ensemble éveille en vous de l'intérêt ou que vous avez déjà pratiqué, sachez que l'architecte de Matrix sommeille peut-être en vous...

Entrée

BTN [*i* [*p*]]
Lit l'état du bouton i pour le joueur p (0 par défaut)
i : 0 à 5 ⇔ gauche, droite, haut, bas, bouton_a et bouton_b

p : joueur 0 or 1
Si aucun paramètre n'est spécifié, renvoie sous la forme d'un mot binaire de deux octets l'état des 12 boutons (P0 : bits 0..5, P1 : bits 8..13)

La correspondance des boutons sur le clavier est :

- Touches du joueur 1 : Touches fléchées Gauche, Droite, Haut, Bas + C/N + V/X
- Touches du joueur 2 : S, F, E, D + tabulation/Shift + A/Q

BTNP [*i* [*p*]]
Identique à **btn()** mais prend la valeur true seulement lorsque le bouton n'a pas été pressé à la dernière image. **btnp()** retourne également true toutes les 4 images après que le bouton a été maintenu pendant 12 images.

Pratique pour des choses telles que « Pressez un bouton pour continuer » ou des déplacements dans un menu.

KEYCONFIG
Donne accès à l'écran de configuration du clavier pour modifier l'affectation des touches.

Audio

SFX *n* [*channel* [*offset*]]
Joue l'effet n sur la canal channel (0 à 3) à partir de la note offset (0 à 31)
n -1 pour arrêter le son sur le canal
n -2 pour sortir le son de la boucle sur le canal
Toute musique jouée sur le canal sera arrêtée.

offset dans les notes (0 à 31)
channel -1 (par défaut) pour choisir automatiquement un canal qui n'est pas utilisé

Cela n'écrasera pas la lecture de la musique. Si rien n'est joué, vérifiez le channel_mask utilisé avec music(). La lecture ne monopolise pas tous les canaux (ex. 1+2 pour le premier canal).

MUSIC [*n* [*fade_len* [*channel_mask*]]]
Jour la musique à partir du motif n (0..63)
n à -1 stoppe la musique
fade_len en ms (0 par défaut)
channel_mask indique quel canaux sont utilisés pour jouer la musique

Ex. pour jouer sur le canaux 0..2 : 1+2+4 = 7
Fixé à 15 par défaut – tous les canaux (1+2+4+8)

IMPORT *nom-du-fichier.wav*
EXPORT *nom-du-fichier.wav*
Importe/Exporte les sons à partir/à destination d'un fichier (sons.wav).

Très pratique pour récupérer des graphismes ou les travailler dans un logiciel externe

Map

MGET *x y*
MSET *x y v*
Lit ou écrit la valeur (v) en x, y.

MAP *cel_x cel_y sx sy cel_w cel_h* [*layer*]
Dessine la portion de la carte (en cellules) à la position sx, sy de l'écran (en pixels).

Si la couche layer est spécifiée, seules les cellules avec le même indication sont dessinées (valeur binaire, ainsi 0x5 provoque le dessin des acteurs avec les bit0 et 2 positionnés ⇔ à true), en l'absence de cette indication, dessine tous les acteur de la portion.

Exception : le acteur 0 n'est jamais dessiné car il correspond à une position vide

Ex. : **map(0, 0, 20, 20, 4, 2)** -> dessine un bloc de 4x2 cellules en 0, 0 dans la carte à la position 20, 20 de l'écran.

Mémoire

PICO-8 dispose de 3 types de mémoire :

1. La mémoire RAM de base (32Kio), voir le schéma ci-après. Accessible à l'aide des fonctions peek(), poke(), memcpy(), memset().
2. La mémoire ROM de la cartouche : schéma indique à la mémoire de base jusqu'en 0x4300. reload() copie la mémoire ROM de la cartouche dans la mémoire de base.
3. La mémoire RAM de LUA (256Kio) : programme compilé + variables. *N'accordez aucune attention à l'homme derrière le rideau.*

Note technique : Vous n'avez probablement aucun intérêt à connaître ceci.

Lors de l'emploi de l'éditeur, les données modifiées sont dans la mémoire ROM de la cartouche, mais les fonctions API comme spr() et sfx() n'agissent que sur la mémoire de base. PICO-8 copie automatiquement la ROM de la cartouche dans la RAM dans deux cas :

1. Lorsque le programme de la cartouche est lancé par run()
2. Lorsqu'un mode d'édition est quitté (pour passer à un autre ou revenir à l'invite de commandes)

Schéma de la mémoire de RAM de base

0x0000 gfx

0x1000 gfx2/map2 (zone partagée)

0x2000 map

0x3000 gfx_props

0x3100 song

0x3200 sfx

0x4300 Laissé à la libre disposition de l'utilisateur

0x5e00 données persistantes de la cartouche (256 octets)

0x5f00 état du dessin (+ non utilisé)

0x5f40 état matériel

0x5f80 GPIO (128 octets)

0x6000 écran (8Kio)

L'encodage des couleurs dans des zones gfx (feuille de acteur) et écran utilise un octet pour stocker 2 pixels. Les bits de valeur faible se situent à gauche.

Le format de la carte (zone map) est d'un octet par cellule (normalement utilisé comme index de acteur).

PEEK *addr*
POKE *addr val*
Lit et écrit un octet à l'adresse addr de la mémoire RAM de base.

L'adressage va de 0x0000 à 0x7fff.

Lire ou écrire en dehors de cette plage provoque une erreur.

MEMCPY *dest_addr source_addr len*
Copie len octets de la mémoire RAM de base à partir de l'adresse source_addr vers l'adresse dest_addr. Les portions peuvent se chevaucher.

RELOAD *dest_addr source_addr len* [*nom-du-fichier*]
Identique à **memcpy** mais s'applique à partir de la mémoire ROM de la cartouche.
La section de code (dont l'adresse est supérieure ou égal à 0x4300) est protégée et ne peut être lue. Si le *nom-du-fichier* est précisé, les données sont chargées à partir d'une cartouche différente.

CSTORE *dest_addr source_addr len* [*nom-du-fichier*]
Identique à **memcpy**, mais copie de la mémoire RAM de base vers la mémoire ROM de la cartouche

cstore() équivaut à **cstore(0, 0, 0x4300)**

Peut être utilisé pour écrire des outils de construction de cartouche ou pour visualiser l'état de la carte ou de la *Feuille des acteurs* utilisée par l'*Editeur*.

Si l'adresse de destination dest_addr est supérieure ou égale à 0x4300, peut écraser le programme sous forme ASCII de la cartouche.

MEMSET *dest_addr val len*
Fixe la valeur de len octets à partir de l'adresse dest_addr à la valeur val.

Très rapide – peut être utilisé pour dessiner des lignes de scan écrêtées, etc.

Math

MAX ***x y***
MIN ***x y***
MID ***x y z***
Renvoie le maximum, minimum et la moyenne des valeurs en paramètres.
Par exemple, **`mid(7, 5, 10)`** renvoie 7.

SGN ***x***
Renvoie l'argument du signe de x : -1 si x <0, 1 sinon.

Si vous avez l'habitude (du fait de l'emploi d'autres langages) d'avoir en retour : -1 si x<0, 1 si x > 0 et 0 si x==0, vous pouvez utiliser l'astuce suivante :

```
FUNCTION IIF(C,T,F)
 IF (C) THEN RETURN T
 ELSE RETURN F
 END
END
V=1
PRINT(IIF(V==0,0,SGN(V)))
```

FLR ***x***
Renvoie l'entier le plus proche de x // x-(x%1)

`FLR(4.1)` renvoie 4
`FLR(-2.3)` renvoie -3.0

COS ***x***
SIN ***x***
Renvoie le cosinus de x, 1.0 correspond à un cercle complet (⇔ 2*PI radians ⇔ 360 degrés).
Le sinus est inversé en conformité à l'espace de l'écran (y croit vers le bas).
Ex. **`sin(0.25)`** renvoie -1 (⇔ sin(360° x 0.25)=1, soit -1 après inversion de l'axe Y).

Si vous préférez des fonctions trigonométriques basées sur les radians sans inversion de l'axe y, vous pouvez redéfinir COS et SIN de la façon suivante :

```
COS1=COS
FUNCTION COS(ANGLE) RETURN COS1(ANGLE/(3.1415*2)) END
SIN1=SIN
FUNCTION SIN(ANGLE) RETURN SIN1(-ANGLE/(3.1415*2)) END
```

ATAN2 ***dx dy***
Convertit dx, dy en un angle de 0 à 1.

Comme pour cos() et sin(), l'angle est pris dans le antihoraire de l'espace de l'écran.
Ex. **`ATAN(1,-1)`** renvoie 0.125 (360° x 0.125 = 45°).

`SQRT x`
Renvoie la racine carrée

`ABS x`
Renvoie la valeur absolue (positive) de x.

`RND x`
Renvoie un nombre pseudo-aléatoire n, qui vérifie 0 <= n < n
Si vous souhaitez un nombre entier, utilisez **`flr(rnd(x)).`**

`SRAND x`
Fixe la graine du générateur de nombre pseudo-aléatoire.
La graine est automatiquement définie au démarrage de la cartouche.

Opérations sur bits

```
BAND x y
BOR  x y
BXOR x y
BNOT x

SHL  x y
SHR  x y
```

Les rotations de bit (SHL et SHR) sont des rotations logiques (le bit de signe n'est pas affecté).

Entrées de menu personnalisé

`MENUITEM index [label callback]`
Ajoute une entrée supplémentaire au menu pause. L'index doit être compris en 1 et 5. Il détermine l'ordre dans le lequel les entrées sont affichées. Le libellé (Label) peut faire au plus 16 caracrtères. Callback est une fonction appelée lorsque l'entrée est choisie par l'utilisateur.
Exemple : **`MENUITEM(1,"Recommencer",function() rejouer() sfx(10) end)`**

Chaîne de caractères

```
S = "THE QUICK BROWN FOX"
-- LONGUEUR
PRINT(#S) --> 19


-- concaténation de chaînes
PRINT("THREE "..4) --> "THREE 4"


-- sub() pour extraire une sous-chaîne
PRINT(SUB(S,5,9)) --> "QUICK"
PRINT(SUB(S,5))   --> "QUICK BROWN FOX"
```

Données de la cartouche

Chaque cartouche est capable de stocker 64 nombres (256 octets) de données persistantes.
Exemple : mémorisation de high score ou sauvegarde de la progression du joueur.

CARTDATA ***id***

Appeler cette fonction après le chargement d'une cartouche. L'id est une chaine d'identification de 64 caractères de long qui doit être suffisamment originale pour éviter à deux cartouches d'avoir le même identifiant.

Exemple : **`CARTDATA("ZEP_JELPI")`**

Les caractères autorisés sont a à z, 0 à 9 et souligné (_).
Renvoie true si l'identifiant a pu être actualisé.
`cartdata()` ne peut pas être appelé plus d'une fois par exécution de la cartouche

DGET ***index***

Renvoie le nombre stocké à la position d'index 0 à 63 (de la zone persistante).
Utilisable uniquement après avoir utilisé préalablement **`cartdata()`**

DSET ***index value***

Définit le nombre stocké à la position d'index 0 à 63 (de la zone persistante).
Utilisable uniquement après avoir utilisé préalablement **`cartdata()`**

GPIO

General Purpose Input Ouput (entrées/sorties à usage général) permet aux machines de communiquer ensemble. PICO-8 associe les octets de la plage 0x5f80 à 0x5fff avec les broches du GPIO qui peuvent être écrites (pour émettre une valeur, par ex. allumer une led) ou lues (par ex. lire l'état d'un interrupteur). Le GPIO a plusieurs significations selon les plates-formes hôtes.

- CHIP .....0x5f80 à 0x5f87 => xio-p0 à xoi-p7

- Pocket CHIP.....0x5f82 à 0x5f87 => GPIO1 à GPIO6 (xio-p0 et p1 sont accessible à l'intérieur du boîtier dans la zone de prototypage)

- Raspberry Pi .....0x5f80 à 0x5f87 => wiringPi broche 0 à 7

Les valeurs CHIP et Raspberry Pi sont toutes TOR (Tout Ou Rien), soit : 0 (niveau bas => 0v) ou 255 (niveau haut => présence tension).

IMPORTANT : pour avoir accès aux broches du GPIO, vous devez lancer PICO-8 avec les droits root : sudo pico8

Un programme simple pour faire clignoter toutes les leds connectées :

```
T=0
FUNCTION _DRAW()
  CLS(5)
  FOR I=0,7 DO
    VAL=0
    IF (T%2<1) VAL=255
    POKE(0x5f80+I,VAL)
    CIRCFILL(20+I*12,64,4,VAL/11)
  END
  T+=0.1
END
```

Une cartouche exporté en HTML5 (html+.js) utilise un tableau global d'entiers (pico8_gpio) pour représenter les broches du GPIO. L'enveloppe HTML doit définir le tableau :

var pico8_gpio = Array(128) ;

Coroutines

Les coroutines permettent l'exécution en parallèle de plusieurs fonctions.

COCREATE *f*

Lance l'exécution d'une fonction en arrière-plan (coroutine) et renvoie son descripteur.

CORESUME *co*

Transfère le contrôle d'exécution à la fonction d'arrière-plan (co).

COSTATUS *co*

Renvoie le statut de la fonction d'arrière-plan (co) :

- **SUSPENDED** ➔ suspendue à réactiver à l'aide de **CORESUME(co)**
- **DEAD** ➔ terminée

YIELD

Rend le contrôle d'exécution au programme principal.

Exemple

```
CO=COCREATE(FUNCTION()
 FOR I=1,10 DO
  PRINT("CO:"..I)
  YIELD()
 END
END)
FUNCTION _DRAW()
 IF COSTATUS(CO)=="SUSPENDED" THEN
   CORESUME(CO)
 END
END
FUNCTION _UPDATE()
END
```

Programmes

D01.p8

```
LOCAL B
FUNCTION _INIT()
 B=0
END
FUNCTION _DRAW()
 LOCAL I,N
 I,N,R=16384,B,""
 CLS()
 CURSOR(0,0)
 WHILE I>=1 DO
  IF N>=I THEN
    R=R.."1"
    N-=I
  ELSE
    R=R.."0"
  END
  I/=2
 END
 PRINT(R)
END
FUNCTION _UPDATE()
 B=BTN()
END
```