PICOLOOP 0.77 README

Manual and tutorial for PC/PSP/PSVita/Pocketchip can be fetch here :

PC / PSP / PSVita / Pocketchip向けのマニュアルとチュートリアルはこちらから入手できます:
https://github.com/farvardin/picoloop-manual

Overview

Picoloop is a synth and a stepsequencer (a clone of the famous nanoloop). The step sequencer play 4 tracks simultaneously, by default each track are 16 steps long, you can set the track to be up to 128 steps. Each track has a synthengine which can be a Virtual Analog, a Drum Synth, a FM synth. Each of the step can be edited on the 4 tracks. So for example, you can set a C4 note on step 0, nothing on step 1 and 2, then a D+5 on step 3 until the step 15.

Picoloopはシンセサイザーであり、ステップシーケンサー（有名なnanoloopのクローン）です。 ステップシーケンサーは4トラックを同時に演奏します、デフォルトで各トラックは16ステップの長さです、最大128ステップになるようにトラックを設定することができます。 各トラックにはシンセジンが備わり、バーチャルアナログ、ドラムシンセ、FMシンセとなります。 各ステップは4トラックで編集できます。 たとえば、ステップ0にC4の音符を設定し、ステップ1と2には何も設定せず、ステップ3のD + 5をステップ15まで設定できます。

A pattern of 16 steps is played repeatedly while these notes can be edited in various respects like volume, pitch, filter, lfo etc. All parameters are set step-wise, so you can make huge variation of sound on the same pattern. Each channel's patterns can be saved to file slots and are then available for new combinations. Finally, saved patterns can be arranged in a song structure.

16ステップのパターンが繰り返しPlayされますが、これらのノートは音量、ピッチ、フィルター、LFOなどをそれぞれにエディットできます。すべてのパラメーターは段階的に設定されるので、同じパターンに対して、さまざまなバリエーションを作ることができます。 各チャンネルのパターンはファイルスロットに保存して、さらに新しい組み合わせに使用できます。 最後に、保存したパターンをソングの構造に配置することができます。

Besides the song editor and manual loading of patterns, there are 3 different ways to modify each channel's patterns so that they form longer structures:

ソングエディタと手動でのパターンのLoadの他に、各チャンネルのパターンをモデファイしてより長い構造をつくる方法が3つあります。

- 1/2, 1/4 or 1/8 speed
- Different pattern lengths by track
- 1/2, 1/4 または 1/8 speed
- トラックごとに違う長さのパターン

Picoloop is targeted for linux/SDL compatible system. It uses SDL to render graphics. It uses RtAudio or SDL to render audio out. It uses RtMidi to send and receive midi message, today only clock

PicoloopはLinux / SDL互換システムを対象としています。 SDLを使いグラフィックスをレンダリングします。 オーディオをレンダリングするためにRtAudioまたはSDLを使います。 これはRtMidiを使ってMIDIメッセージを送受信しますが、今はMIDI clockだけです。

How it works

There is two pane :
2つのペインがあります:

- the 16 steps ;
- 16ステップ ;
- the menu which is above the 16 steps ;
- 16ステップ以上のメニュー;

On the 16 boxes, there is a moving cursor which displays the current played step. And another cursor which displays the current selected step. There is a text menu, which allows you to select channel and edit parameter. All notes in the sequencer can be edited simultaneously.

16個のボックスには、現在再生中のステップを表示する移動カーソルがあります。 mまた、それとは別に現在選択されているステップを表示するカーソル。 チャンネルを選択してパラメータを編集するためのテキストメニューがあります。 シーケンサー内のすべてのノートを同時に編集できます。

The menu mode

The menu mode changes the selected textual menu at the bottom of the screen. This allows you to choose what kind of parameter you want to change, for example, the

NOTE, the OSC, the Filter... In menu mode, when you press left/right, it changes the textual menu in the bottom of the screen

メニューモードは、画面下部の選択されているテキストメニューを変更します。 これにより、どの種類のパラメータを変更するかを選択できます。例えば、NOTE、OSC、Filter ...メニューモードで、左または右を押すと、画面下部のテキストメニューが変わります。

```
</>             move backward/forward in the menu
^/v             select the track from 0 to track 3
B               do nothing
A + </>/^/v,    edit all steps at once
A               enter the edit mode
L               go to the next     16 steps on the same track
R               go to the previous 16 steps on the same track
SELECT          call menu2 or menu1 ( it cycle between the two menu )
```

The edit mode

The edit mode changes the current step on the 16 steps display. In edit mode, when you press left/right, it changes the current step in the sequencer.

編集モードは16ステップディスプレイ上の現在のステップを変更します。 編集モードで左または右を押すと、シーケンサーの現在のステップが変わります。

```
</>             select next step
^/v             select step+4, step-4
B               enter a note in a step, it works as a cut/paste
A + </>/^/v,    edit the current step
A               do nothing
L               go to the next     16 step on the same track
R               go to the previous 16 step on the same track
SELECT          call menu2 or menu1 ( it cycle between the two menu )
START           cycle between the different parameter
```

menu1

The menu 1 is the first menu. It allows to modify the parameter of the synthengine of the current track. Each synth engine has different kind of parameter.

メニュー1が最初のメニューです。 現在のトラックのシンセエンジンのパラメータを変更することができます。 各シンセエンジンには異なる種類のパラメータがあります。

You can:

1. A/R : the enveloppe of the synth
2. Note : the value of the note trigged
3. OSC : the oscillator of the synth
4. VCO : the vco parameter for example a osc1 and osc2 mixer
5. LFO : the lfo parameter
6. FLTR : the filter parameter.

Here is the global overview of the most common parameters you will find by synth engine.

シンセエンジンにおける最も一般的なパラメータの全体的な概要です。

- A/R :
    - Amp Enveloppe
        - Attack/Release Amp / FM operator1
        - Decay/Sustain Amp / FM operator1
    - Filter Enveloppe
        - Attack/Release Filter / FM Operator2
        - Decay/Sustain Filter / FM Operator2
    - Trig/Amplification
        - trig time/Volume
- Note :
    - Choose note with "C3" "C4" "D2"
    - Choose note with "dot"
- OSC :
    - Choose the two OSC shape of the synth, the drumsynth, the FM synth
- VCO :
    - VCO mix in synth mode, OP1mult/OP2mult in FM mode
    - OP1amp/OP2amp in FM mode
- LFO :
    - depth/speed in psynth and drumsynth mode
    - pitchbenddepth/pitchbendspeed in psynth mode
- FLTR :
    - Cutoff/Resonance

- algo/mode (lp/bp/hp)

menu 2

The menu 2 is a more general menu. You can:

メニュー２はより一般的なメニューになっています:

1. L/S load and save a pattern track or the 4 pattern track ;
2. BANK change the current bank which allow you to have more pattern.
3. PSH shift left or right the pattern of a track and add bunch of 16 step ;
4. MAC change the current synth engine, synth, drumsynth, dbopl (adlib fm)
5. FX apply fx on the current track delay
6. BPM change the BPM and swing of the 4 track and the step diviser of the current track

L/S menu

The top menu shows the tracks and patterns.

トップメニューにはトラックとパターンが表示されます。

- Save your current track by selecting an empty slow, then B+down.
- Save your current pattern (group of 4 tracks) by selecting a column, then A+down.
- Load a track with A+up (it will replace your current working track).
- Load a pattern with B+up (il will replace the 4 current working tracks).

You can clone a track by loading it into memory, then saving it to another track.

トラックをメモリにロードして別のトラックに保存することで、クローンを作成できます。

The menu below (song position) allows you to order your tracks. You can enter it with Start key.

下のメニュー（ソングのなかの位置）では、トラックを操作することができます。 スタートキーで入力できます。

- Define Loop start position with A+up
- Define Loop end position with A+down
- Change the values with B+up and B+down.

PSH menu

- Increase the track size (16 steps block) with A+UP
- Decrease the track size (16 steps block) with A+DOWN
- Increase the track size (1 step) with A+RIGHT
- Decrease the track size (1 step) with A+LEFT
- Navigate into the new steps (move the position by -16 or +16) with the LEFT and RIGHT SHOULDER (or TAB and BACKSPACE on PC)

What you need to know

Every pattern is stored in the bank/bankXXX/file so if you backup this directory you should be fine.

すべてのパターンはbank / bankXXX /ファイルに保存されているので、このディレクトリをバックアップすれば大丈夫です。

You can write pattern on OpenDingux and push it on your laptop, unfortunately, it could sound little different. The OpenDingux a320 lack FPU so every thing is done in fixed point. On PC and PSP it's done in floating point…

あなたはOpenDinguxでパターンを書いてあなたのラップトップにそれをプッシュすることができます。しかし残念ながら、それは少し違ったように聞こえるかもしれません。 OpenDingux a320にはFPUがないため、あらゆることは固定小数点で実行されます。 PCとPSPでは、浮動小数点で行われます。

You won't be able to push pattern from PC to PSP/Opendingux, it could work but it depend if the Machine is available on the PSP. Today (picoloop 0.75c) only Picosynth, Picodrum, DBopl and PBSynth are available on slower platform : PSP, opendingux.

あなたはPCからPSP / Opendinguxにパターンをプッシュすることができないでしょう、それはうまくいく場合もあるかもしれません、しかしそれはマシンがPSPで利用可能であるかどうかによります。 今現在（picoloop 0.75 c）より遅いプラットフォームで利用可能なのは、Picosynth、Picodrum、DBopl、およびPBSynthだけです：PSP、opendingux

The combo Select+UP, Select+Down allow you to raise or decrease the internal volume of the apps

Select+上、Select+下のコンボを使用すると、アプリの内部音量を上げ下げすることができます

If you plug on Windows/Linux a midi cable you can use it with midi sync in or midi sync out. The MMC message ("play", "stop") are received by picoloop, but it doesn't transmit this kind of message. It understand the "stop" and "play" MMC message.

Windows / LinuxにMIDIケーブルを接続すると、MIDI同期入力またはMIDI同期出力で使用できます。 MMCメッセージ（ "play"、 "stop"）はpicoloopによって受信されますが、この種のメッセージは送信されません。 MMCメッセージの「停止」と「再生」を理解しています。

You can use midi delay to fine tune the sync between midi hardware and picoloop software.

あなたは、MIDIハードウェアとpicoloopソフトウェアの間の同期を微調整するためにMIDI遅延を使うことができます。

The MDA Drum synth works with file patch ".ds" file. So if you have a laptop with a harddrive which is not an ssd, you should work with a PSU and not the battery to avoid disk spinning down. The automatic disk shutdown will block audio for 10ms and it became really audible. Will be fixed in a latter version.

MDAドラムシンセはファイルパッチ ".ds"ファイルで動作します。 あなたはssdではないハードドライブを搭載したラップトップを持っているのであれば、ディスクの回転ダウンを避けるために、バッテリーではなく、PSUで作業する必要があります。 自動ディスクシャットダウンは10msの間オーディオをブロックし、それは本当に聞こえるようになりました。 後者のバージョンで修正される予定です。

Package dependency, build from source

This section describes the dependency for building picoloop. On most platform in 2016, you will need this kind of package.

このセクションでは、picoloopをビルドするための依存関係について説明します。 2016年のほとんどのプラットフォームでは、この種のパッケージが必要になります。

```
- libsdl*      => version 1.2 with the "ttf" and "gfx"
 - libasound2*  => alsa
 - libpthread*  => pthread for multiple thread
 - g++          => I use the g++ 4.7.2
 - make         => I use the make 3.81
 - libjack-dev  => I use jack 2.0, and the jack dependency is needed today
```

LASTEST SOURCE

You can always update to the lastest source. But, you should then switch to a tag.

あなたはいつでも最新のソースにアップデートすることができます。 しかし、その場合はタグに切り替える必要があります。

```
# git clone https://github.com/yoyz/audio.git

# git tag

# git checkout picoloop-0.75b
```

Tag are safier. And today, there is a known issue, the build of PatternPlayer need two build. The issue come from the "mkdir" which is launch after the other command, so every thing seem to fail on the first make... Unfortunately I have not fixed this issue because you fall into this issue only when cloning the software from the repo.

タグは安全です。 そして今日、既知の問題があります。PatternPlayerのビルドには2つのビルドが必要です。 問題は他のコマンドの後に起動される "mkdir"から来るので、すべて最初のmakeで失敗するようです...残念なことにあなたがリポジトリからソフトウェアをクローンするときだけこの問題が起こるので私はこの問題を修正できていません 。

COMPILATION FOR DEBIAN

```
# cd      picoloop
 # make -f Makefile.PatternPlayer_debian_RtAudio clean
 # make -f Makefile.PatternPlayer_debian_RtAudio
```

COMPILATION FOR POCKETCHIP

You have to build it on the pocketchip.

```
# cd      picoloop
 # make -f Makefile.PatternPlayer_pocketchip_RtAudio clean
 # make -f Makefile.PatternPlayer_pocketchip_RtAudio
```

COMPILATION FOR RASPBERRYPI1

You have to build it directly on the raspberry. If you have a raspberry 2 or 3, you should go to the "debian" section. This build type is a light version like the opendingux one.

ラズベリーパイの上に直接構築する必要があります。 あなたがラズベリーパイ2または3を持っているならば、あなたは「debian」セクションに行ってください。 このビルドタイプはopendinguxビルドタイプのようなライトバージョンです。

```
# cd      picoloop
 # make -f Makefile.PatternPlayer_raspi1_RtAudio    clean
 # make -f Makefile.PatternPlayer_raspi1_RtAudio
```

COMPILATION FOR OPENDINGUX

On debian you need the /opt/opendingux-toolchain/ directory. you can fetch fetch it there

Debianでは /opt/opendingux-toolchain/ ディレクトリが必要です。あなたはそれをfetchすることができます

: http://www.treewalker.org/opendingux/

```
# cd      picoloop
 # make -j 64 -f Makefile.RtAudio_opendingux               clean
 # make -j 64 -f Makefile.PatternPlayer_opendingux_RtAudio clean

 # make -j 64 -f Makefile.RtAudio_opendingux
 # make -j 64 -f Makefile.PatternPlayer_opendingux_RtAudio
 # make -j 64 -f Makefile.PatternPlayer_opendingux_RtAudio
```

COMPILATION FOR GP2X

On debian you need the /opt/open2x/ directory. you can fetch it there :

debianでは /opt/open2x/ ディレクトリが必要です。 あなたはそれをそこにfetchすことができます：

http://wiki.gp2x.org/articles/i/n/s/Installing_the_Open2x_toolchain.html I don't have build it month ago, so at your own risk.

```
# cd      picoloop
 # make -j 64 -f Makefile.RtAudio_opendingux               clean
 # make -j 64 -f Makefile.PatternPlayer_opendingux_RtAudio clean

 # make -j 64 -f Makefile.RtAudio_opendingux
 # make -j 64 -f Makefile.PatternPlayer_opendingux_RtAudio
 # make -j 64 -f Makefile.PatternPlayer_opendingux_RtAudio
```

COMPILATION FOR WINDOWS WITH MINGW32

On debian, you need the mingw32 package. Picoloop provide the header and binary for SDL 1.2 on windows. So here it is a "cross build" you build on linux for windows.

Debianでは、mingw32パッケージが必要です。 PicoloopはWindows上でSDL 1.2のヘッダとバイナリを提供します。 だからここでは、Linux上にWindows向けにビルドする "クロスビルド"です。

```
# cd          picoloop
 # make        -f Makefile.PatternPlayer_windows_mingw_RtAudio          clean
 # make        -f Makefile.PatternPlayer_windows_mingw_RtAudio
```

COMPILATION FOR PSP

you need to have the sdk, the toolchain and the external lib. All can be fetched here :

SDK、ツールチェーンと外部ライブラリが必要です。 すべてここで取得することができます：

https://github.com/pspdev

```
# cd      picoloop
 # source ~/local/pspdev/env_build
 # make -f Makefile.PatternPlayer_psp_SDL        clean
 # make -f Makefile.PatternPlayer_psp_SDL
```

LAUNCHING

```
# ./PatternPlayer
```

KEYS

Debian/Windows

```
- ESC                : Select key : go back to the global menu/switch menu
- ENTER              : Start  key : change sub menu
- L-CTRL             : A      key : insert note/delete node/copy note
- L-ALT              : B      key : change the value on the screen of the
selected step
- ESC+ENTER          : quit
- TAB                : LEFT Shoulder
- BACKSPACE          : RIGHT Shoulder
```

Depending on the current menu selected : A/R, OSC, VCO, BPM, LS, etc.

現在選択されているメニューに応じて： A/R, OSC, VCO, BPM, LS, etc.

```
- L-ALT + up          : change value up
- L-ALT + down        : change value down
- L-ALT + left        : change value left
- L-ALT + right       : change value right
```

OpenDingux/PSP/GP2X

```
- SELECT              : enter menu and move between menu
- ENTER               : Start key/enter a menu
- A key               : insert note/delete node/copy note
- ESC+ENTER           : quit
```

Depending on the current menu selected : A/R, OSC, VCO, BPM, LS, etc.

現在選択されているメニューに応じて：A/R, OSC, VCO, BPM, LS, etc.

```
- B key + up          : change value up
- B key + down        : change value down
- B key + left        : change value left
- B key + right       : change value right
```

USAGE

file named dataP<%d>T<%d>.pic store information about saved pattern. This file will be created when you store file in the "L/S" menu.

dataP<%d>T<%d>.pic という命名されるファイルに保存されたパターンに関する情報を格納します。 このファイルは「L/S」メニューにファイルを保存したときに作成されます。

DEBUG

You need two file font.bmp and font.ttf which are located next to the source.

ソースの隣にある2つのファイルfont.bmpとfont.ttfが必要です。

DEBUG PSP

You need your psp gcc compiler, the library and psplinkusb.

あなたのpsp gccコンパイラ、ライブラリ、そしてpsplinkusb。

1. Launch psplink on the psp this is an EBOOT.PBP.
2. as root, launch usbhostfs_pc (you need to be root to access the psp by usb and you need to be in the somewhere/psplinkusb/usbhostfs_pc/ folder). You should see it is connected.
3. In another terminal launch pspsh. Try to 'ls -l', to see if all is ok. Copy the prx file in the usbhostfs_pc directory, you should see it with 'ls -l'

If you want to launch PatternPlayer.prx, launch it with pspsh : ./PatternPlayer.prx In this case, you need the prx file (not an elf or EBOOT.PBP).

PatternPlayer.prxを起動する場合は、 pspsh : ./PatternPlayer.prx を使用して起動します。この場合、prxファイルが必要です（elfやEBOOT.PBPは不要です）。

If you want to debug it with psp-gdb, you can do it by this command in two different window.

psp-gdb でデバッグしたい場合は、2つの異なるウィンドウでこのコマンドを使用して実行できます。

```
    host0:/> debug PatternPlayer.prx
```

In another window :

もう一方のウィンドウで：

```
    $ psp-gdb ./PatternPlayer.elf
 (gdb) target remote localhost:10001
     Remote debugging using localhost:10001
```

In this case you need a prx and an elf file.

この場合、prxファイルとelfファイルが必要です。

DEBUG PS VITA

read https://tai.henkaku.xyz/ you need to go to this website, then use the vita exploit on your ps vita at this url http://henkaku.xyz/go/ Here your ps vita should have the "beta exploit". You need to have psp2shell compiled from https://github.com/yoyz/psp2shell it's a from from https://github.com/Cpasjuste/psp2shell but it should be at the right commit level. Compiled psp2shell so you have a psp2shell binary Compiled the kernel and user module and put it on your ps vita in ux0:/tai Then build picoloop for your psvita with -DDEBUG_PRINTF, it should work

https://tai.henkaku.xyz/を参照し、あなたはこのウェブサイトに行く必要があります、そしてこのurl ://henkaku.xyz/go/ であなたのps vitaで” vita exploit ”を使う必要があります。

ここであなたのps vitaは " beta exploit "が必要になります。

https://github.com/yoyz/psp2shell から” psp2shell “をコンパイルする必要があります。これは、https://github.com/Cpasjuste/psp2shell からのものですが、正しいコミットレベルである必要があります。

“ 得るためにPsp2shell ”バイナリを得るために” psp2shell “をコンパイルしました。カーネルとユーザーモジュールをコンパイルし、それをあなたのps vitaの ux0:/data (ディレクトリです)に置きます。

その後、-DEBUG_PRINTを使用してps vita用のpicoloopをビルドします。これでうまくいくはずです。

grab your core file in the ux0:/data directory, the last core

最後のコアである ux0:/data ディレクトリにあるコアファイルを取得します。

use vita parse core from here : https://github.com/xyzz/vita-parse-core install it as root using the pip as root :

ここからvita parse coreを使用してください: https://github.com/xyzz/vita-parse-core

rootとしてインストールします。root権限でpipを使用します:

pip install -r requirements.txt

then $ zcat psp2core-1510435186-0x0000092601-eboot.bin.psp2dmp > psp $ python /home/peyrardj/build/vita-parse-core/main.py psp picoloop_vita.elf

You can also push the eboot.bin binary directly in the good location with this command

このコマンドでeboot.binバイナリを正しい場所に直接プッシュすることもできます

psp2shell> put eboot.bin ux0:/app/PICOLOOP1/

Changelog

V0.77abcd

- a : introduce lgptsampler for vita and raspberry pi
- b : introduce lgptsampler for windows, stuck on psp
- c : fix the crash on clang++, need to fix all int myfunc() { } without return 0; in the bracket...
- c : lots of valgrind
- c : move to RtAudio 4.1.2 to 5.0
- c : update to lastest psp2shell on vita
- d : avoid an insidious graphic bug on vita on display_graphic_loadsave()
- d : avoid a weird behaviour in lgpt on vita with resonance which crash the app when it is tweaked
- d : this version work on opendingux, psvita, windows, fail on psp

V0.76abc

- a : first port on psvita thanks to #henkaku team
- b : re add polyrithmic in PSH,
- b : fix bpm speed (slightly higher)
- b : STOP the sequencer with A+B in BPM and START the sequencer wih A+START in BPM
- b : fix ps vita LTRIGGER and RTRIGGER
- b : fix the SELECT+UP/Down to change volume, but not change the track
- b : MAX_PATTERN_BY_PROJECT update to 128 for all platform, but not PSP, opendingux, gp2x
- c : introduce a SELECT+RIGHT|LEFT to speed up reduce the tempo on the main menu a little like elektron box

V0.75abcdef:

- remove a lot of duplicate code in XXXUserInterface, because it was really needed

- a add the resid engine
- b export DUMP_AUDIO=1 now export a wav file
- c add SDL2.0 compatibility and keep SDL1.2 as default because it works

V0.74acd:

- add pocketchip platform, which is a "kind of variant" of raspberry pi 1, thank to garvalf from chipmusic.org
- fix a PatternElement.cpp issue related to a value not initialized
- shift track to the left and to the write in the song "orderlist"
- fix a bug with the note tuning which appear from 0.71 to 0.74

V0.73 :

- add a midi delta option in the [BPM] menu to move the sync signal in time, it is a kind of "midi delay" but doesn't work exactly the same way
- add a 'osc scale' which allow to change oscillator one or oscillator two pitch to make basic chord with two note
- rework a bit BPSynth on opendingoo to allow to remove some bad FixedPoint behaviour
- fix first buffer audio generation of pbsynth which lead to an audio glitch

V0.71 :

- improve realtime of mda drum synth, now quite useable, but need to be improved

V0.70 :

- add a readme.html in the repo
- fix some glitch on the screen from 320x240 to 640x480

V0.69 :

- add midi in sync on linux/windows
- add midi out sync on linux/windows
- midi on windows add issue with latency
- windows can run MDADrumSynth, still with issue with the latency