v3.0 User's Manual

Table of Contents

Controls

At any time, Select can be pressed to get help.

Basic Usage

Dpad moves the cursor up/down/left/right (wraps around edges)
X places a hit. Press again to make an accented hit
O erases a hit, or turns an accented hit into a regular hit
Left/Right trigger = change pattern
Triangle + left/right trigger = change tempo (BPM)
Start = start/stop pattern playback
Square + left trigger = copy a pattern
Square + right trigger = paste a pattern
Triangle + dpad right = toggle track mute
Triangle + dpad left = toggle track solo
Analog stick up + right trigger = copy pattern to next empty pattern
Dpad left/right + left/right trigger = accelerated cursor movement
The analog stick moves from screen to screen:
Analog down = move to lower screens (Song, Sounds, Notes, Files)
Analog up = move to pattern editor
Analog left/right = move between screens (wraps around)
Files <=> **Sounds** <=> **Song** <=> **Notes** <=> **Files**

Sample Modification

These functions work in almost every screen.
If the cursor is on a note, the changes affect only that step:
    Square + analog up/down = change current step's volume
    Square + dpad up/down = change current step's pitch
If the cursor is not on a note, the changes are pattern-wide:
    Square + analog up/down = change current sample's volume
    Square + analog left/right = change current sample's pan
    Square + dpad up/down = change current sample's pitch
    Square + circle = reset current sample's volume, pan and pitch

Notes Screen

Dpad left/right = move cursor left/right
Dpad up/down = raise/lower note by half-step, if cursor is on a note
Dpad up/down = move from track to track, if cursor is not on a note
Square + dpad up/down = raise/lower current note by octave
Triangle + left trigger = delete step, slide rest of track to the left
Triangle + right trigger = insert note, slide rest of track to the right
Triangle + dpad up/down = change tracks, even if cursor is on a note
Square + analog up/down = change step volume
Square + left trigger = copy note with volume
Square + right trigger = paste note with volume
Circle on empty step = preview sample
Left/Right trigger = change pattern, following the song playback cursor if song is playing
Left/Right trigger = change pattern, following the song edit cursor if song is not playing
Dpad left/right + left/right trigger = accelerated pattern switching (+/- 5 patterns)

Sounds Screen

O = preview sample
X = change sample
While changing samples:
- Dpad left/right = select samples
- Dpad up/down = change sample directories
- X = keep the new sample
- Triangle = back out the change
- You can also use Square controls to modify pitch, pan and volume

Song Editor

The bottom row of numbers show the position in the song
The top row of numbers shows the pattern number to be played at each step
Dpad left/right = forward/backward through the song
Dpad up/down = change pattern number at that position
X = set song start position
O = set song end position
Analog stick down + start = start song playing where edit cursor is
Left/right trigger = scroll through song, update pattern at top
Square + left trigger = copy pattern number
Square + right trigger = paste pattern number
Triangle + left trigger = del pattern from song
Triangle + right trigger = insert pattern into song
Dpad left/right + left/right trigger = accelerated cursor movement

Files Menu

Dpad = navigate menus
X = execute menu item

Load song - scans songs directory for *.kick files.
Up/down to scroll through them
X = load song
Triangle = cancel

Save as .kick - saves pattern, song, soundpack, and skin in a .kick file
X = save
Left/right = move cursor
Up/down = change letter
Square = delete letter
Triangle = back out

Save pattern as .wav - saves current pattern to patterns/songname_patternxx.wav
Save song as .wav - saves entire song as CD quality .wav file to songs/songname.wav
Save song as .mid - saves entire song as .mid file to songs/songname.mid

Load sound pack - scans soundpacks dir for *.pack files.
Up/down = scroll through names
X = load it
Triangle = cancel load

Load skin - scans through "skins" dir for *.skin files.
up/down = scroll through names
X = load it
O/triangle = cancel load

Soundpacks

Soundpacks are stored in the soundpacks/ directory.
Each soundpack has a filename called *.pack, e.g.
rel100.pack, avtomat.pack, shifty.pack, nullsleep.pack
Each *.pack file has a list of drum numbers and sample names:

```
drum01 kick/kickburnt.wav
drum02 noise/goodsmash.wav
drum03 hat/zighat.wav
```

Notice each samplename is comprised of a directory name (e.g. kick/) and a samplename (e.g. kickburnt.wav). It's very important to keep them organized this way, because you can **Audition Samples:** cursor around on the sample names and change drum sounds. You can move on top of a snare, hit X, then left/right to hear all of the different snares. Pressing up/down changes from snare to another type of drum. When you find what you like, hit X again to load it and keep writing your song. Or O to cancel loading. This feature would not work if the directories were not organized.

All of the samples are stored in `soundpacks/drums/`
Inside `soundpacks/drums/` are a number of directories for organizing the samples.
e.g.

```
/Volumes/Untitled/PSP/GAME/KDEV
1/soundpacks/drums>ls -l
total 1088
drwxrwxrwx  1 carbondi  admin  768 24 Jul 22:58 bass/
drwxrwxrwx  1 carbondi  admin  32768 26 Jul 14:51 beep/
drwxrwxrwx  1 carbondi  admin  32768 26 Jul 14:51 bell/
drwxrwxrwx  1 carbondi  admin  32768 26 Jul 14:51 clap/
```

and inside each of those dirs are a number of samples for your soundpack-making pleasure.
e.g.

```
KDEV/soundpacks/drums//clap:
total 448
-rwxrwxrwx  1 carbondi  admin   79596  8 Jul 12:21 808Clap.wav*
-rwxrwxrwx  1 carbondi  admin  127736 26 Jul 14:51 emptyclap.wav*

KDEV/soundpacks/drums//cymbal:
total 576
-rwxrwxrwx  1 carbondi  admin   38900 24 Jul 22:58 cymdist.wav*
-rwxrwxrwx  1 carbondi  admin  127692 26 Jul 14:51 nervouscymbal.wav*
-rwxrwxrwx  1 carbondi  admin   83920 18 Jul 18:15 revcymbal.wav*
```

BTW samples must be <= 128KB in size or they'll get truncated!
They must also be 44.1KHz 16-bit, stereo Windows PCM .wav samples or they will sound too fast or too slow.

Making Good Kits

Making good kits is an art. Here are some critical points:
1) Play with the kit and see how well it works.

2) Make sure all the samples sound approximately the same volume. Samples with high-pitched content have to be smaller in amplitude (quieter) than others. E.g. Kick samples can reach the top and bottom of your sample editor window, but snares should only be about half that height. Hi-hat samples also should be a little quieter than kicks.

3) Make sure the order of the samples makes sense. So far, there has been a convention to put kicks at the top, followed by snares below, followed by hi-hats, then toms, then whatever. Weird stuff usually goes in the bottom row. If you follow this convention it'll make it easier for people to use your kit and **especially to use your kit with different songs**.

4) Consider including multiple types of some instruments. One of the most important sounds in a kit is the snare. To a large extent, it determines the style of music. If you include 2 or 3 snares in your kit, then it becomes a flexible kit.

5) Play with the kit and see how well it works. Don't release it until it sounds good in a song. (Yes, this is the same as #1).

Releasing a kit/Installing a Kit

For maximum smoothness, there is a way to package your kit that makes it simple for everyone to install.

What you want to do is make it so that users can simply unzip your kit into the **`soundpacks/`** directory to make everything work.

In order to do this, you must organize your files like this:

1) Make a directory and put your `whatever.pack` file in it.

2) Make a directory in that one called `drums/`

3) Inside `drums/` make a directory for each type of sound your kit has.
e.g.

```
drums/kick/
drums/snare/
drums/hat/
drums/noise/
```

4) Follow this example, to edit your `whatever.pack` file to look like this:

```
# drum samples by shifty http://gweep.net/~shifty/snackmaster

drum01 kick/sludgekick.wav
drum02 kick/abruptkick.wav
drum03 snare/whacksnare.wav
drum04 snare/happysnare.wav
drum05 snare/heavisnare.wav
drum06 hat/carlclosed.wav
drum07 hat/openhatshing.wav
drum08 tom/buzztom.wav
drum09 clap/emptyclap.wav
drum10 rim/ringingrim.wav
drum11 cymbal/nervouscymbal.wav
drum12 beep/gridbeep.wav
```

Note that comments are encouraged. Go ahead and put the URL you designed in there.. Just start the line with a #:

5) After you've edited your `pack.whatever file`, create the zip.

From unixy systems, do:

```
zip -r whatever.pack.zip *
```

From win32, create a recursive zip of all files in the same directory where `whatever.pack` and `drums/` are located.

6) If your zip file has a directory structure like this, then you did it right:

```
~/psp/kick/src/pack.shifty>unzip -v shifty.pack.zip
Archive:  shifty.pack.zip
 Length   Method    Size  Ratio   Date   Time   CRC-32    Name
--------  ------  ------- -----   ----   ----   ------    ----
       0  Stored        0   0%  07-25-05 22:59  00000000  drums/
       0  Stored        0   0%  07-25-05 22:59  00000000  drums/beep/
   77424  Defl:N    66943  14%  07-25-05 22:33  fa8f4d1b  drums/beep/gridbeep.wav
  127108  Defl:N    93824  26%  07-25-05 22:50  52649ffd  drums/beep/highbeep.wav
       0  Stored        0   0%  07-25-05 22:59  00000000  drums/bell/
  127164  Defl:N   103132  19%  07-25-05 22:52  8e15392e  drums/bell/spookowbell.wav
       0  Stored        0   0%  07-25-05 22:59  00000000  drums/clap/
  127736  Defl:N    96768  24%  07-25-05 22:50  e530edc5  drums/clap/emptyclap.wav
       0  Stored        0   0%  07-25-05 22:59  00000000  drums/cymbal/
  127692  Defl:N   105371  18%  07-25-05 22:51  a65f3fec  drums/cymbal/nervouscymbal.wav
       0  Stored        0   0%  07-25-05 22:59  00000000  drums/hat/
   51176  Defl:N    38719  24%  07-25-05 22:20  5a8893c6  drums/hat/carlclosed.wav
   60964  Defl:N    46324  24%  07-25-05 22:39  ae4b2781  drums/hat/openhatshing.wav
       0  Stored        0   0%  07-25-05 22:58  00000000  drums/kick/
   96956  Defl:N    80090  17%  07-25-05 22:11  bcb98cb6  drums/kick/abruptkick.wav
  127312  Defl:N   107550  16%  07-25-05 22:52  e2a55a94  drums/kick/brzzkick.wav
   62392  Defl:N    58373   6%  07-25-05 22:36  9872dcd7  drums/kick/carlkick.wav
   92040  Defl:N    74381  19%  07-25-05 21:50  357acb2d  drums/kick/hardkick.wav
  102600  Defl:N    76931  25%  07-25-05 22:40  13c128ca  drums/kick/houseykick.wav
  118152  Defl:N    95899  19%  07-25-05 22:40  611de84c  drums/kick/lowkick.wav
  125840  Defl:N   113008  10%  07-25-05 22:49  967ead74  drums/kick/marykick.wav
  127492  Defl:N   104473  18%  07-25-05 22:53  eb4e10b0  drums/kick/sludgekick.wav
       0  Stored        0   0%  07-25-05 23:00  00000000  drums/noise/
   98668  Defl:N    66764  32%  07-25-05 22:40  ef3286f2  drums/noise/panping.wav
  120420  Defl:N    95888  20%  07-25-05 22:47  fb01019f  drums/noise/tinsel.wav
       0  Stored        0   0%  07-25-05 22:59  00000000  drums/rim/
  113320  Defl:N    84706  25%  07-25-05 22:13  f1998469  drums/rim/ringingrim.wav
       0  Stored        0   0%  07-25-05 22:58  00000000  drums/snare/
   83572  Defl:N    69017  17%  07-25-05 22:07  f5229958  drums/snare/80ssnare.wav
  127016  Defl:N    92372  27%  07-25-05 22:51  b02f731f  drums/snare/bratsnare.wav
   84316  Defl:N    70879  16%  07-25-05 22:35  4f8fe097  drums/snare/brightsnare.wav
  127876  Defl:N   106834  17%  07-25-05 22:49  28b0117e drums/snare/desolatesnare.wav
  127080  Defl:N   103898  18%  07-25-05 22:51  c9ba67d6  drums/snare/fryingpansnare.wav
   87156  Defl:N    67083  23%  07-25-05 22:39  51842d90  drums/snare/happysnare.wav
  127152  Defl:N    91739  28%  07-25-05 22:50  b25450ed  drums/snare/heavisnare.wav
   70740  Defl:N    57262  19%  07-25-05 22:41  99a4f28d  drums/snare/opensnare.wav
   97588  Defl:N    74501  24%  07-25-05 22:12  6d40f197  drums/snare/whacksnare.wav
       0  Stored        0   0%  07-25-05 22:59  00000000  drums/tom/
   87524  Defl:N    68707  22%  07-25-05 21:53  4d1417da  drums/tom/bendtom.wav
  127284  Defl:N   118784   7%  07-25-05 22:48  a3d7d3da  drums/tom/buzztom.wav
     389  Defl:N      204  48%  07-26-05 14:48  d16985a8  shifty.pack
--------          -------  ---                            -------
 3034299          2430906  20%                            42 files
```

Notice how there's a `shifty.pack` file in there, and everything else is inside the `drums/` directory.

Also notice that if you create your own, new, wild directory name in the zip file, that's perfectly okay! Unzip will create those new directories on your PSP and put all the files in.

Graphics Graphics Graphics

Most graphics in PSPKick are stored in files and loaded at runtime. We think this is better than storing graphics inside the program because it allows the program to be more easily customized. Most digital graphic formats (like png and jpeg) are complicated to read. Therefore, to get the software out more quickly, PSPKick uses its own simplified file formats called .vrm and .avr instead. To make skins, you'll use tools to convert them. For most graphics, you'll use CtrlView (by A. Matveev at www.ctrlview.com) to translate from .jpeg into .c, then picpack (by the PSPKick team) to go from .c into .vrm. For other graphics, you'll use Super Flavor, or "max.php" (originally gfxConverter by Alonetrio of WAV, modified by the PSPKick team for alpha channel and binary output) to go from .png to .avr.

.vrm Files

.vrm means "Video RaM." You convert from jpeg to .c, then from .c to .vrm with picpack. .vrm means simply a list of pixels in two-byte format (big endian). The bits in each 16-bit word go: 0BBBBBGGGGGRRRRR. .vrm files do not have dimensions in them. In other words, the size of the image is not included.

.avr Files

.avr means "Alpha Video Ram." They're simply 32-bit files that go ARGB, 4 8-bit values. Like .vrm, they contain no dimension information. You can't tell what size they're going to be.

Using PicturePacker

picpack03 is a dead simple command line utility that turns .c files into .vrm files. Typically .c files are generated from a tool that turns jpegs into .c files.
The .c files have lots of lines that look like: RGB(255,127,128),

To use picpack in win32, you must place cygwin1.dll
(if you don't have it, google for it) in the same dir as picpack02.exe.
Then type:

```
picpack03 screenOverlay.c screenOverlay.vrm
```

That will read in screenOverlay.c and write out screenOverlay.vrm.
It will also print out some useless debug info.

Using picpack in unixy environments is easy, too. just type:

```
./picpack03 screenOverlay.c screenOverlay.vrm
```

Using Super Flavor, or "max.php"

Super Flavor is easy to use. Just install php with the gdImage library on your computer (wr/t the OS X distribution, the install may not set your path correctly. try /usr/local/php5/bin/php). Then type:

```
/usr/local/php5/bin/php max.php terminal_font_9
```

That will turn terminal_font_9.png into terminal_font_9.avr.

Making a Skin

A skin is a 480x272 graphic. It has a region for entering notes, called the Upper Panel, and a region for changing samples, using menus, and editing the song. it's called the Lower Panel. The upper panel has a subsection where the drumhits and cursor work. Its size is 448x168. There are 12 rows of 32 squares in that region. Each square in that region is 14x14. The cursor and drumhit graphics are all 14x14.

There are actually 4 different lower panel menus. They all have the same size. It's still changing, but at the moment, it's 330x54 .

Splash Screen

It's not stored with the skin. It's in the gui/ dir. Its name is hardcoded: splash02.vrm
It's size is the full size of the screen: 480x272.

Help Screens

They are not included in the skins directories.
They are not listed in the skin.*files.
Their filenames are hardcoded.
There are stored in the gui/ directory.

```
gui/helpWindowPatt.vrm
gui/helpWindowSong.vrm
gui/helpWindowSampName.vrm
gui/helpWindowCustomScr.vrm
```

Their size is fixed at 448*168.

Params you can change in *.skin files

Skin files are stored in the skins/ directory. They must have the filename format: rel100.skin, supershifty.skin, tatooed.skin, etc. In addition to the *.skin filename, there must be a directory inside skins/ with the name of the skin. For example, if the skin filename is skins/rel100.skin, theall of the files for the skin go into skins/rel100/. Each skin stores all of its files inside its own directory. There is no (simple) way to use pieces of one skin in another one.

Here is an example file, rel100.skin. Note, the green parts are just comments in this manual to help you understand it, they should NOT be included in your .skin files.
They are comments here to explain what each param does.

```
# upper panel
background screenOverlay06.vrm    This is the name of the 480x272 image
topX 24        this is where on the screen it begins drawing the 32x12 note grid
topY 32

# cursor                                All cursor files are 14x14.
cursorEmptyLit celEmptyLit.vrm        filename of the .vrm where the cursor is stored.
cursorFilledLit celFilledLit.vrm           the cursor and a regular hit together
cursorAccentLit pacwithcherries.vrm        the cursor and an accented hit together
cursorFilled celFilled.vrm                    A regular hit with no cursor
cursorAccent cherries.vrm                     An accented hit with no cursor

# pattern/tempo font
numbsfont numbers03.vrm           this is the font name for the numbers used in temp and BPM.
numbsFontW 7                               width of letters
numbsFontH 9                          height of letters
numbsFontVwid 105                       width of jpeg/png/vrm file in pixels
numbsFontVhei 121                         height of jpeg/png/vrm file in pixels
numbsFontHsp 14         hor. space between letters in jpeg/png/vrm.  includes letter and emptiness
numbsFontVsp 16         vert. space b/t letters in jpeg/png/vrm.  includes letter and emptiness
numbsFontGridW 8        number of letters in each row of the jpeg/png/vrm
numbsFontSpace 7        how much space to put between letters when drawing with it in menus, etc.

# tempo/pattern locations
pattNumX 142                       Location of the top, left corner of the pattern # display
pattNumY 16                        (be sure to leave space to the right for three digits)
tempoX 396                         Top, left corner of the BPM display
tempoY 16

# chaser
chaserX 16                              Top, left location of the chaser gfx
chaserY 31
chaserGFX ghost.vrm                  the graphics file to use for the chaser gfx
chaserW 14                           the size of the chaser gfx
chaserH 14

# traffic light
greenLight greenLightOn.vrm       filename of greenlight on graphic.  dimensions below
greenLightX 4                                position on screen of greenlight gfx
greenLightY 251
greenLightW 15                               size of greenlight gfx in pixels
greenLightH 15
yellowLight yellowLightOn.vrm      same for yellow light
yellowLightX 4
yellowLightY 239
yellowLightW 15
yellowLightH 15
redLight redLightOn.vrm                 same for red light....
redLightX 4
redLightY 225
redLightW 15
redLightH 15

# lower panel
lowerPanelX 24           Position of the lower panel on the screen
lowerPanelY 208
lowerPanelW 330        Size of the lower panel
lowerPanelH 54

sampleNameBG bluebar.vrm      Filename for the sampleName screen image
songEditorBG greenbar.vrm                  same for song editor.  these screens must all have
```

```
customScreenBG purplebar.vrm             the same dimensions as the lowerPanelW and lowerPanelH
mixerBG bluebar.vrm                         (mixerBG isn't used...yet!)

# terminal font
termFont termfont.vrm      name of the font file used for sample names and menus.  lots of params
termFontW 7                              width of letters
termFontH 9                           height of letters
termFontVwid 105                        width of jpeg/png/vrm file in pixels
termFontVhei 121                          height of jpeg/png/vrm file in pixels
termFontHsp 14            hor. space between letters in jpeg/png/vrm.  includes letter and emptiness
termFontVsp 16            vert. space b/t letters in jpeg/png/vrm.  includes letter and emptiness
termFontGridW 8                    number of letters in each row of the jpeg/png/vrm
termFontSpace 7           how much space to put between letters when drawing with it in menus, etc.

# layout of sample names page
sampX 20                        x,y = where *on the lower panel* the names start (top left corner)
sampY 7
sampTileX 100            the horizontal spacing for the three columns
sampTileY 10              the vertical spacing for the four rows

# layout of song editor
songX 50                          x,y = where *on the lower panel* the song editing stuff starts
songY 8
songTileW 24                    How far apart the columns are
songTileH 10                     How far apart the rows are

# layout of custom screen
custLoadX 25        x,y of load and save are where those menus begin *on the lower panel* (top,
left  corner)
custLoadY 2
custSaveX 156
custSaveY 2
custTileH 10                           vertical spacing between names in the menus
custTileW 10                           (custtilew is not actually used- no use for it)
custFileRqX 25                      the x,y coords of where the filename requestor pops up
custFileRqY 40
```

Font Details

To make your own font:
Lay out the following grid in an anti-aliased font of your style and any size (start with about 8-12 pixels). Notice it's 8 rows of letters high and 8 columns of letters wide.

```
  ! " # $ % & '      ( note first char is space, ' is an apostrophe)

( ) * + , - . /      (, is comma)

0 1 2 3 4 5 6 7

8 9 : ; < = > ?

@ A B C D E F G

H I J K L M N O

P Q R S T U V W

X Y Z [ / ] ^ _    (last char is underscore)
```

The space between each char should be wide enough that anti-aliasing trails from e.g. the the K don't interfere with the neighboring letters J, L, C and S. So, put as much space as you want - just as long as they don't overlap. That might be 8 pixels or so.

The color should be whatever color you want the font to be. The background should be black.

output: must be .png file with an 8-bit alpha channel. the .png should be the kind of .png that has variable alpha channel, not just on/off. The .png file gets converted with max.php, which is a modified and ehhanced version of gfxConverter by AloneTrio of WAB.

Font Dimensions

There are two sets of spacing:
First, the size of the letters themselves. e.g. the last ones were 9x11.
Second, the the spacing between where each letter starts in the grid. e.g. if the letters were 9x11, and you used six pixels of buffering, the spacing would end up 15x17. I realize this is pretty anal, but it makes our program look great.

PSPKick Music File Formats

These file formats were created just for PSPKick. They're all in ASCII, so that they are easy to read. Note they are also in DOS text format, which means lines are ended with carriage return, linefeed, a.k.a. CR/LF, a.k.a. 0x0d 0x0a, a.k.a. "\r\n", a.k.a. 13 10.

master.txt

This file is read at startup to determine the initial skin, the initial song, and a few other things. It is currently saved from the Files Menu, when you save song as a .kick file:

```
currentSong sweater        The last song you were editing

skin rel300.skin           The current skin

curEditPatt 10            The pattern you were editing when you last saved

end master
```

songs/sweater.kick - Drum section

The songs directory is filled with .kick files. Each one contains a PSPKick song, minus the samples. It contains the sample names in a song, the song information and the pattern information.

```
version 3.00

song song
start 01
end 10
drum01 synth/koto.wav
drum02 synth/koto.wav
drum03 hat/OPHAT.WAV
drum04 snare/snare2.wav
drum05 hat/909Openhat.wav
drum06 clap/808Clap.wav
drum07 tom/Lowtom.wav
drum08 tom/HighTom.wav
drum09 clap/808Clap.wav
drum10 rim/707Rimshot.wav
drum11 synth/synthHit.wav
drum12 kick/abruptkick.wav
```

songs/sweater.kick - Song section

```
song song
start 01
end 16
songstep 01 01 10 01 10
songstep 02 02 10 01 10
songstep 03 10 10 01 10
songstep 04 03 10 01 10
songstep 05 02 10 01 10

...

songstep 68 01 10 01 10
songstep 69 01 10 01 10
endsong
```

Where "start 01" indicates that the first step in the song is 1,
and "end 16" means the last step in the song is 16.

songs/sweater.kick - Patterns Section

```
patt 01
tempo 14300
numSteps 32
numTracks 12
drmStat01 070 128 128
drmStat02 070 128 128
drmStat03 065 128 128
drmStat04 074 128 128
drmStat05 100 128 128
drmStat06 070 128 128
drmStat07 070 128 128
drmStat08 070 128 128
drmStat09 070 128 128
drmStat10 070 128 128
drmStat11 070 128 128
drmStat12 091 128 128
trk 01 243c 0 0 0 0 0 0 0 0 243c 0 0 0 0 0 0 0 0 243c 0 0 0 0 0 0 0 0 1f3c 0 0 0 0 0 0 0 0
trk 02 0 0 0 0 0 273c 0 0 0 0 0 0 0 0 1d3c 0 0 1d32 0 0 1d26 0 1f3c 0 0 0 0 0 0 0 0 223c 0 0 0
trk 03 243c 0 0 0 0 0 0 0 0 0 0 0 0 243c 0 0 0 0 0 0 0 0 0 0 0 0 243c 0 0 0 0 0 0 0 0
trk 04 0 0 0 0 0 0 0 0 243c 0 0 0 0 0 0 0 0 0 0 0 0 243c 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
trk 05 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
trk 06 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
trk 07 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
trk 08 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
trk 09 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
trk 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
trk 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
trk 12 243c 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 243c 0 0 0
mute 2 3 4 12
solo 5 6
endpatt
```

In the drum parameters section "drmStat":

The three columns following each drmStat are the volume, pan and pitch for each sample. For each one, 128 is the middle. For volume, the range is from 0 to 255. 0 is about -48dB, 255 is about +12 dB. For pan, the range is 0 to 255. 0 is all the way left, 255 all the way right. For pitch, the range is 29 to 227. 29 means -100 cents from normal. 227 means +100 cents from normal. Cents are 1/100ths of an octave. So, the range is +/- 1 octave, with an absurd amount of precision. (Hint, pitch two channels one cent apart for chorus!)

The first number after "trk " is the track number. The next 32 numbers after it are step values. The step value has both a note number and a volume encoded into it. It uses hexadecimal encoding. The lowest 8 bits are the volume (only values 0-100 are allowed). 0 is the quietest, 100 is the loudest. A typical hit is 60. An accented hit is 72. The next 8 bits are the note number. Only values 0-73 are allowed. 36 represents middle C. For example, "2a3c" means the volume is 0x3c, or 60. The note is 0x2a or 42, meaning F#4.

The tempo is stored in 1/100ths of a BPM. So, 13300 = 133.00 BPM.
If you needed 125.67 BPM, it would be "tempo 12567"

If you are very clever, like one user, you can change the numSteps to e.g. 24.
Some functions may not work well, and it's not officially supported, so don't do it! :)

Don't increase the number of tracks beyond 12. It's never gonna work! :)

Appendix A - Release Testing

With all of these great features, how to test them in a reliable way? Here are all the things to test:

Cursor movement:

- from zone to zone
- in pattern editor zone
- in song editor zone
- in "custom" (load/save) zone
- in sample name zone

Controls:

- pattern editor zone
  - tempo, shift pattern, change pattern, copy patt, paste patt
- song editor zone
  - tempo, change pattern, copy patt num, paste patt num, change patt, ins patt, del patt
- sample name zone

Customization:

load several soundpacks w/ music playing

- do all gfx elements change?
  - font
  - background
  - lower menu panels

load several skins w/ music playing

- does each and every sample change?

Start/stop playback:

- in pattern zone
- in song zone

Modify sample params:

- volume
- pan
- pitch

Internal file formats:

- modify patt data, save, exit, reload, verify
- modify song data, save, exit, reload, verify
- modify sample stats (pitch/vol/pan), save, exit, reload, verify

Export file formats:

- sound quality means length, volumes, tempo, pitch, etc.
  - export pattern .wav, verify filename and sound quality
  - export song .wav, verify sound quality
  - export song .mid, verify playback in several programs

Appendix B – Credits

Noah Vawter – Programming and Manual
Nathan Wray – Graphics and Public Relations

"If we have been able to see far, it is because we have stood on the shoulders of giants."-paraphrased Albert Einstein.

PSPKick was initially based on the Hello World and PSPSnd programs by nem, and on the OldskOOl intro by Alonetrio of WAB, Rex^syndicate and Jester^sanity.

It was built using one of the earlier versions (20050613) of the ps2toolchain by the ps2dev gang (marcus, ooPo, etc.)

Screenshot.bmp code was supplied by weak from Austria (I'm still looking for that sidescrolling shooter! There's been a dirge of them by SomeONe You know! call up pimpot, ok?) Update: I think this program became Callisto! Very beautiful, the boss was real hard! Looking forward to the next update.

Thanks to maxconsole.net for hosting the PSPKick forums, and ThePSPimp.com for doing it when we first started out.

Thanks to joostp and Guest for helping with dev tools back at the beginning.

Thanks to the random irc users who answered technical questions. Thumbs down to those who wouldn't help with USB unless I was using the sdk.

Thanks to Richard Stallman and the GNU Project for enabling us to use gcc and many other things for absolutely free.

Word up SoCal gang, it's Squarez fault this prg is so late!

Sen Lee>Any questions?
shifty>Yeah, how do you speak Chinese?

Big shout out to Diallo, who created soundpacks, buzz and the intro song for us. Thanks for keeping up with us and spreading the word. We hope you stay for the whole ride!

Thanks to Gadgetboy, Joe Devlin, Nu11s1eep, Shifty, Zod and the Dobox crew for your soundpacks as well!

Huge props to www.pspupdates.com for your continued support in bringing the hype.

And of course, to all the forum users for adding your two cents and showing us love!

Appendix C - Super Flavor

The following code is used to convert .png files to .avr files. This is used for making custom fonts.

```
<?PHP
/*
PNG to .AVR format converter
AVR = 8888 Alpha Red Green Blue
Coded By AloneTrio www.Wab.Com
alpha channel, bin format by shifty 7/28/05, Oakland, CA
thanks to rob g!

usage:  /usr/local/php5/bin/php max.php terminal_font_9
will read terminal_font_9.png and turn it into terminal_font9.avr
*/

$img = $argv[1];

$im = ImageCreateFromPng($img.".png");
$img_high = imagesy($im);
$img_width = imagesx($im);

$file = fopen($img.".avr", "w+");

for($y=0;$y<$img_high;$y++){
  for($x=0;$x<$img_width;$x++){
    $cpt++;
    $rgb = ImageColorAt($im, $x, $y);
    $r = intval(($rgb >> 16) & 0xFF);
    $g = intval(($rgb >> 8) & 0xFF);
    $b = intval(($rgb) & 0xFF);
    $colorrgb = imagecolorsforindex($im,$rgb);
    $a = $colorrgb['alpha'];
    print "$r $g $b $a\n";

    $str = pack("CCCC",$a,$r,$g,$b);
    fwrite($file,$str,4);

  }
}

fclose($file);

?>
```

Appendix D - picpack03

The following program is used to convert .c graphics files into .vrm's, for making skins.

```
#include <stdio.h>


// picpack03 by shifty 2005
//
// a little ditty to parse .c files generated by imageconverter.
// packs the c-style format into binary, which is compact and easily
// readable dynamically inside PSP programs
//
// version 01 used to work with "imageConverter"
// version 02 is for CtrlView.  This kind of format:
// RGB(  0,   0,   1), RGB(  2,   0,   1), RGB(  0,   0,   0), RGB(  0,   2,   0),

//
// version 03 can also do .pnm, which are binary files of rgb values.
//        they start with a simple header, which jpeg tools can convert to,
//        using "jpegtopnm."
// P6
// 014 014
// 255

// should compile trivially with gcc picpack.c -o picpack
// usage:  pickpack03 helpWindowSampName.c helpWindowSampName.vrm

// PNM mode:
// picpack03 -pnm pnm/$names[$i].pnm vrm/$names[$i].vrm 0
//

enum {
  ST_IDLE=0,
  ST_OPEN,
  ST_DONE,
};



int parsePic(char *,char *,int);
int pnmParsePic(char*,char*,int);

unsigned int array[1000][1000];
int width=0,height=0;
int x=0,y=0;

int main(int argc,char *argv[])
{
  int dith=0;

  if( (argc!=3) && (argc!=4) && (argc!=5) ){
    printf("picpack -- packs .c files generated by image converter into binary.\n");
    printf("   usage:  pickpack <inname> <outname> <optional noise mix>\n");
    printf("  e.g.  pickpack screenOverlay.c screenOverlay.vrm\n");
    exit(0);
  }

  switch(argc)
```

```
  {
  case 3:
    parsePic(argv[1],argv[2],0);
    break;

 case 4:
    dith=atoi(argv[3]);
    parsePic(argv[1],argv[2],dith);
    break;

  case 5:   // pickpack02 -pnm wang.pnm wang.vrm <dith>
    dith=atoi(argv[4]);
    pnmParsePic(argv[2],argv[3],dith);
    break;



 }

  printf("picpack  - shifty\n");

  return(0);
}



int parsePic(char *filenamein,char *filenameout,int dith)
{
  char bufin[2],bufout[3];
  FILE *fdin,*fdout;
  int idx=0,done=0;
  char ch;
  int val[3],pack;
  int i,n1,n2,n3;

  int state=ST_IDLE;

  fdin=fopen(filenamein,"r");
  fdout=fopen(filenameout,"wb");

  val[0]=0;

  while(state!=ST_DONE)
  {
    fread(bufin,1,1,fdin);
    ch=bufin[0];


    switch(state)
    {
    case ST_IDLE:
      switch(ch){
      case '[':
      state=ST_OPEN;
      break;
      default:
      break;
      }
      break;

    case ST_OPEN:
```

```
switch(ch){
case '0':
case '1':
case '2':
case '3':
case '4':
case '5':
case '6':
case '7':
case '8':
case '9':
val[0]*=10;
val[0]+= ch-'0';
//    printf("val[0]=%d\n",val[0]);
break;

case ',':     // shift it along
val[2]=val[1];
val[1]=val[0];
val[0]=0;
break;

case ']':
height=width;
width=val[0];
val[0]=0;
printf("w,h = %d,%d\n",width,height);
break;

case '}':
x=0;
y++;
break;

case ')':         // emit symbol. val[0] = 8-bit R, val[1]=8-bit G, val[2]=8-bit B

//    printf("%d %d %d\n",val[0],val[1],val[2]);

/*
// noise mix
for(i=0;i<3;i++){
  n1 = (random()%(dith/2))-(dith/4);
  n2 = (random()%(dith/2))-(dith/4);
  n3 = n1+n2;
  val[i] += n3;
}
*/

// quantize
val[2] = val[2]>>3;
val[1] = val[1]>>3;
val[0] = val[0]>>3;

//pack=(val[0]<<10) | (val[1]<<5) | (val[2]);
pack=(val[2]<<10) | (val[1]<<5) | (val[0]);

array[x][y] = pack;
x++;
idx++;

// prepare for next number
```

```
      val[0]=0;
      break;

      case ';':
      state=ST_DONE;
      break;

      default:
      break;
      }
      break;

    default:
      break;
    }
  }

  // they're in the file upside down, so do this funny way:
  for(y=height-1;y>=0;y--){
    for(x=0;x<width;x++){
      pack = array[x][y];
      bufout[0] = (pack>>0)&0xff;    // least significant
      bufout[1] = (pack>>8)&0xff;    // most significant
      fwrite(&bufout[0],1,1,fdout);
      fwrite(&bufout[1],1,1,fdout);
    }
  }

  fclose(fdin);
  fclose(fdout);
  printf("w,h = %d,%d\n",width,height);
  printf("Converted %d pixels.\n",idx);
  printf("blah\n");
}


int pnmParsePic(char *filenamein,char *filenameout,int dith)
{
  unsigned char bufin[80],bufout[3];
  FILE *fdin,*fdout;
  int idx=0,done=0;
  char ch;
  unsigned int val[3],pack;
  int i,n1,n2,n3;
  char tmp[80],buf[80],buf2[80];
  char *ptr;

 int state=ST_IDLE;

  fdin=fopen(filenamein,"r");
  fdout=fopen(filenameout,"wb");

  fgets(buf,80,fdin);  // P6 means pnm, binary rgb

  fgets(buf,80,fdin); // this one has the width and height in it oh joy
  ptr = strtok(buf," ");
  width=atoi(ptr);
  ptr = strtok(NULL," ");
  height=atoi(ptr);

  fgets(buf,80,fdin);   // 255
```

```
 for(y=0;y<height;y++){
    for(x=0;x<width;x++){

      fread(bufin,1,3,fdin);
      val[2] = bufin[2]>>3;
      val[1] = bufin[1]>>3;
      val[0] = bufin[0]>>3;
      pack=(val[2]<<10) | (val[1]<<5) | (val[0]);

   //      printf("%d %d 0x%06x %d %d %d\n",x,y,pack,val[0],val[1],val[2]);

      array[x][y] = pack;
    }
    //    printf("\n");
  }

 for(y=0;y<height;y++){
    for(x=0;x<width;x++){
      pack = array[x][y];
      bufout[0] = (pack>>0)&0xff;    // least significant
      bufout[1] = (pack>>8)&0xff;    // most significant
      fwrite(&bufout[0],1,1,fdout);
      fwrite(&bufout[1],1,1,fdout);
    }
  }

  fclose(fdin);
  fclose(fdout);
  printf("w,h = %d,%d\n",width,height);
  printf("Converted %d pixels.\n",width*height);
  printf("blah\n");
}
```

Appendix E - batch convert

This is a batch conversion utility that will turn a directory full of .jpg files into a directory full of .vrm files. It's useful for making skins.
It requires having jpegtools installed on your system.

```
#!/usr/bin/perl

# batch convert jpegs into .vrm files for PSPKick skins

@names =("mainscreen_v2",   # main background

         "cursor",              # cursor, empty 14x14
         "cursor_dot","dot",     # cursor on beat.   beat.
         "cursor_accent_dot","accented_dot",   # cursor on accented beat.  accented beat.

         "chaser_light",               # chaser light

         "redLightOn","yellowLightOn","greenLightOn",      # redlight, yellow light, green
light

         "sounds_screen",      # "samples" menu
         "song_screen",        # "song" menu
         "files_screen",       # "custom" menu
         "notesScreenCrude",   # notes menu

         "song_current_pos", "song_cursor", # song indicator, cursor


   );

for($i=0;$i<=$#names;$i++)
{
    $cmd="jpegtopnm jpg/$names[$i].jpg > pnm/$names[$i].pnm";
    print "$cmd\n";
    system($cmd);

    $cmd="../../picpack03 -pnm pnm/$names[$i].pnm vrm/$names[$i].vrm 0";
    print "$cmd\n";
    system($cmd);

    print "\n";
}
```