Notes on SYNTAL(V.00.2) for OSX
Chapter 2: The `vw`, `vw_` and `r` Macros

Wayne Slawson

May 21, 2004

The SYNTAL source file.

All SYNTAL source files must begin with one or more `#include` lines that read in a bunch of definitions of the symbols you will be using to specify your music. The first such line is as follows (the quotation marks are required):

```
#include "/usr/local/bin/qmSYNTAL03.dfs"
```

It may be that this file[1] is in some other directory in your system; on my computer it is in `/usr/local/bin` along with several other files, such as `csp`.

You may write your own `#include` file—let's say it's called `mymacs.dfs` and is in the directory where your SYNTAL source files are going to reside—then after the first line you may `#include ".\mymacs.dfs"`, your macro definitions. On the SYNTAL CD are two files that you may move to your home directory, `mymacs.dfs` and `model.c`. [For safety's sake, always copy these files and change the copies. Don't edit them directly.] As a rule, it's best to make a new directory for each piece (or project) that you work on, and then copy the `mymacs.dfs` and `model.c` files into that new directory, keeping `mymacs.dfs` with the same name and changing `model.c` to the name of some section of your piece. Try out your knowledge of UNIX commands (see Chapter 1) by carrying out this process, ending up in your new directory. It will take four commands.

Let's say the file name of the section of the piece you getting ready to specify is `tune.c`. Your `tune.c` should start off with some comments that identify what it is your doing, etc.—whatever you want. You'll notice some

[1] This is the file for the 2003 version of SYNTAL; subsequent versions may be slightly different—say, `qmSYNTAL04.dfs`.

comments already in `model.c`. They are whatever is typed between `/*` and `*/`. Just be sure whatever comments you type fall between those special marks; anything outside those marks will be treated as if you intended it to be a sound specification.

As the first two lines of `tune.c`—the first two "real" lines that is, which are not comments—you type the `#include` statements (don't forget the quotation marks!):

```
#include "/usr/local/bin/qmSYNTAL2K.dfs"
#include "./mymacs.dfs"
```

The next thing that is required in all SYNTAL source files is a `START` line. This line has an *argument* enclosed in parentheses whose value is the initial metronome mark. It is a number with a decimal point—the decimal point is very important, omitting it is the most common error in writing SYNTAL source files—that represents the duration of the *beat* expressed as the number of beats per minute. All duration values—almost always the first argument of all SYNTAL source statements—are interpreted as beats, not as seconds or some other unit. The statement `START(60.)` sets the tempo to 60 beats per minute and the duration of a beat to 1 second; the statement `START(140.)` sets the tempo to 140 beats per minute and the duration of a beat to about 0.42857 seconds.

The only other statement in a SYNTAL source file that is actually required is the last statement, `END`. Pretty simple, but don't forget it.

If you make a habit of copying some working SYNTAL source file (like `model.c`) when you begin to prepare a new file, then all these required statements will be taken care of more or less automatically, and you don't have to worry about them.

Between the `START` statement and the `END` statement come the specifications of the sounds in your piece.

The `vw` macro.

The best way to start with SYNTAL is simply to jump into the middle of it by learning a particular, very useful, macro, `vw`.

We'll get to that in a minute, but first let's introduce the concept of an *event*. There's nothing particularly complicated here, it's just that it is useful to be able to refer to all the macros with a single term; `vw` is technically a C-language Pre-Processor macro, but it corresponds to what we might call a musical "event". Not everything that one might call a musical event can

be specified by a single-line macro, nor does a single macro always produces something you would want to call a musical event, but the ideal is to try to set things up (as I tried to do in the `qmSYNTAL2K.dfs` file and the `mymacs.dfs` files) so that one-musical-event-per-macro is the general rule.

Alright, now let's start with an example of `vw`:

```
vw( 3.5,AA,EE, FF,  P, MF,EF4,PPP,  Z)
```

This means that we want a sound event that is 3.5 beats in total duration (don't forget the decimal point!). The event is to begin with the *sound color* AA and ends with the sound color EE. The palette of sound colors is shown in Figure 1. The FF (*fortissimo*) in the next parameter means there is to be a fairly loud beginning to the sound; the P (*piano*) in the next parameter means that there is to be an abrupt drop in the loudness right after the loud attack; and the MF (*mezzoforte*) in the next parameter calls for a crescendo over the course of the sound to a medium-loud level near its end.

> (SUGGESTION: Most dynamic marks are three characters or less. You may be tempted to crowd them together, but it's a good idea—it improves readibility—to allow three character spaces for all dynamic marks, filling with leading blanks as in the example above.)

The pitch of the sound—the EF4—is to be the E-flat above middle-C on the piano (a DS3 would be the same pitch an octave lower). The last two parameters control the level of noise that is to be added: the PPP says that we want a low level of noise at the beginning of the event and the Z says that the noise is to drop away to essentially "zero" at the end of the event. That's it. It's a good idea to try out a short piece using just `vw` to get your feet wet at the very beginning.

You need to know that the duration can be almost anything you want, except that it cannot work out to be less than about 0.010 seconds. The dynamic marks range from a high of F4F (*ffff*) down to P7P (*ppppppp*), Nte (*niente* or "nothing"), and Z (zero—actually a little less than Nte) (for a complete list, see Table 1).[2] Pitch symbols are in two parts, first pitch-

[2]The loudness symbols in Table 1 are as they are defined in the file `qmSYNTAL2K.dfs`. The I in the symbols is intended to suggest an "increment" slightly above the "normal" value of any given loudness, the D a corresponding "decrement". The values assigned to the loudness symbols are relative levels in decibels (dB). If you wish, you may define your own loudness symbols in your file "mymacs.dfs". However, if you use values higher than 75.0 you are very likely to generate out-of-range values, and since `Nte` results in inaudibility in nearly all circumstances, values less than 30.0 don't usually make much sense.

class (AF, GN, DS, etc.) and then octave number. The octave number 4 is the one in the middle of the piano keyboard starting with C-natural and ending with the B-natural eleven half-steps above. Pitches are defined to a high pitch of BN7 (the second-highest note on the piano keyboard) and a low pitch of CN0 (the C-natural nine half steps below the piano keyboard). There are a few more ultra-low pitches defined in the file `qmSYNTAL03.dfs`; if you want other pitches—say, Pythagorian tunings—you can define them in your own `mymacs.dfs`.

Some cautionary notes:

1. Almost all alphabetic characters in SYNTAL source files are upper case. (`Nte` is one popular exception.)

2. It's possible to use numbers instead of the alphabetic symbols, but it's better to use symbols. Remember, if you don't like the ones provided in SYNTAL, you can define your own in your `mymacs.dfs` file.

The `vw_` and `r` macros

Now it's easy to go on to another macro that is closely related to `vw`, namely `vw_`. They look alike; only the `_` is different. It says: "Make a `vw` event with a rest after it."

The specifications are also similar. Let's start with the example of `vw` above and turn it into a `vw_`:

```
vw_( 0.35,AA,EE, FF,  P, MF,EF4,PPP,  Z,0.25)
```

The only difference is the `,0.25` at the end. What it means is that (as before) the total duration of the event is 0.35 beats, but now the *sound* is only 0.10 beats long and it is followed by a silence 0.25 beats in duration. Be careful to make the rest at the end shorter than the total duration at the beginning—SYNTAL doesn't like negative durations.

You could get the same thing as `vw_` with two macros:

```
vw( 0.10,AA,EE, FF,  P, MF,EF4,PPP,  Z)
r(  0.25)
```

The `r` macro is simply a rest. It has only a single argument: the duration of the rest in beats. You might say that in this case two macros together define a single event, whereas in `vw_` the ideal, "one macro per event", is achieved. Different strokes for different folks, however; you can do it either way.

Remember the durations in nearly all macros are all expressed in units of beats. And the duration of a beat is determined by the `START` macro at the beginning of the piece. This makes it handy to adjust tempos without having to change the durations of a whole bunch of events.

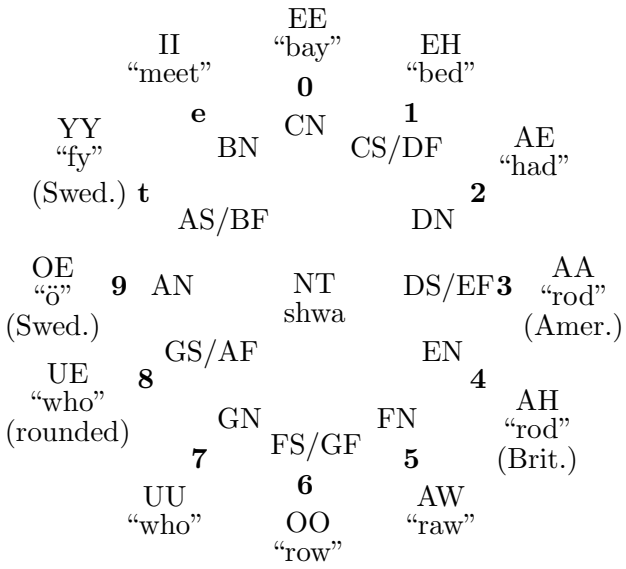

Figure 1: The Sound Colors in SYNTAL
The sound-color classes are presented in a circle, each illustrated by a word containing a vowel in the associated sound-color class.

Table 1: Loudnesses in SYNTAL
Note: The loudness symbol **Z** has the relative level of 0 dB. In most cases **Nte** is advised to avoid anomalously low levels.

Symbol	Level (in dB)	Symbol	Level (in dB)
F5F	75.	P	57.
F5D	74.5	PD	56.
F4I	74.	PPI	55.
F4F	73.5	PP	54.
F4D	73.	PPD	53.
F3I	72.5	P3I	52.
FFF	72.	PPP	51.
F3F	72.	P3P	51.
F3D	71.	P3D	50.
FFI	70.	P4I	49.
FF	69.	PPPP	48.
FFD	68.	P4P	48.
FI	67.	P4D	47.
F	66.	P5I	46.
FD	65.	PPPPP	45.
MFI	64.	P5P	45.
MF	63.	P5D	44.
MFD	62.	P6I	43.
MPI	61.	P6P	42.
MP	60.	P7P	39.
MPD	59.	P8P	36.
PI	58.	Nte	30.