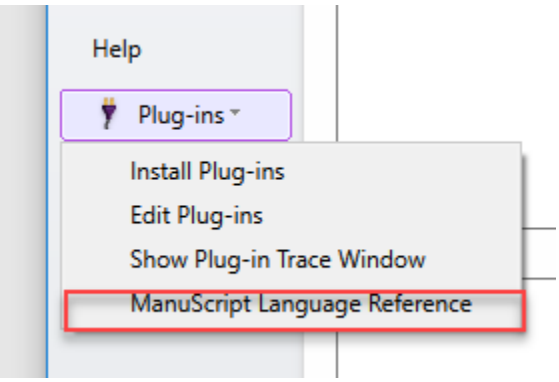
Case study: writing a very simple plugin: Hide All Voices Except 1

Bob Zawalich 20 December 2020

Sometimes you want to automate a relatively simple process for which there is no published plugin. If you are comfortable with using a macro processor like *Keyboard Maestro* or *AutoHotkey*, you might set up a macro for yourself.

In some cases you could instead write a plugin to do this. While published plugins, like those that ship with Sibelius, or those you can install from the plugins download page, need to be general purpose, and need to handle input errors and be clearly documented, a single-use plugin that only you will use has few of those requirements.

You will need some familiarity with the ManuScript language to write a plugin. The language reference is available in Sibelius at File>Plug-ins>ManuScript Language Reference.



But for this example, I will show you some code that will accomplish a task, and demonstrate how to turn it into a plugin.

There is a fairly detailed and illustrated writeup up on using *New* in the plugin dialog editor, so I direct you to http://www.bobzawalich.com/wp-content/uploads/2020/02/Write-your-own-simple-plugin-1.pdf to see the details.

Here is the problem to solve:

"Is there a way/plugin to make this happen in basically one click:
Show voice 1 in selection and hide all other voices.
Show voice 2 in selection and hide all other voices.
Show voice 3 in selection and hide all other voices."

If I were going to write such a plugin for publishing, I would likely put up a dialog where you could choose the voice to show, and maybe have an option to mute the hidden objects as well.

But let's aim at simplicity, and assume you will have a plugin that will hide all voices except one that you specify, and that you will make 3 copies of the plugin to deal with other voices.

I would write some code that looks like this:

```
voice = 1;
for each obj in Sibelius.ActiveScore.Selection
{
   if (obj.VoiceNumber != voice)
   {
      obj.Hidden = True;
   }
```

}

What does this do?

- I set the variable *voice* to be 1, so that I leave voice 1 alone. I could just hard-code *1* in the code, but having a variable will make it easier when I make copies of the plugin.
- *for each obj in Sibelius.ActiveScore.Selection* looks at each selected *Bar Object* in the current score. *Bar Objects* include notes, lines, and text, and most things that can be placed in a bar.
- When I get an object *(obj),* I look to see is its *VoiceNumber* property matches what I have assigned to the variable *voice*, which in this case is 1. If it does not, (*!=* means does not equal), then we hide it, by setting the *Hidden* property to True.

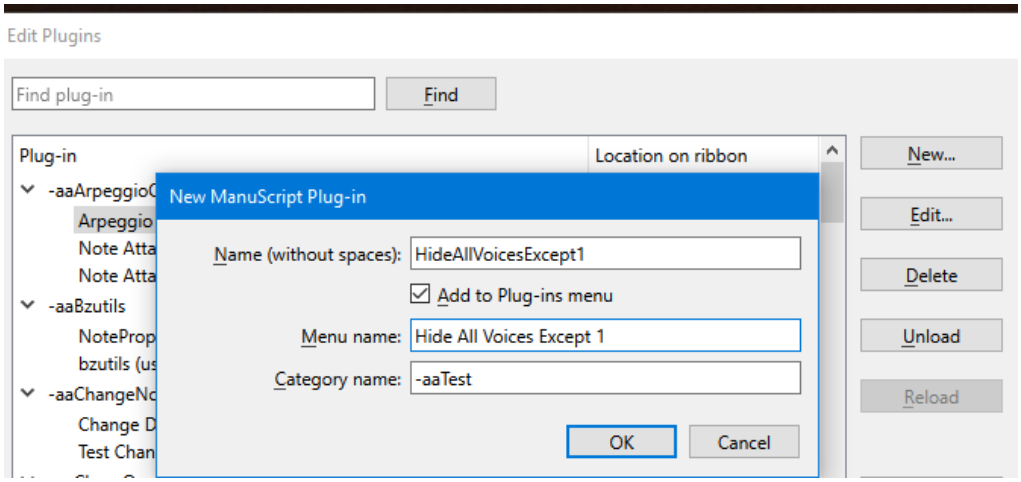
So this matches the original specification. It does not mute the object *obj*, and it does not select the changed objects, which are things I might do in a published plugin. I does not warn you if there is no selection, or check if certain objects cannot be hidden. It does not do different things in scores and parts.

But it does what you asked it to do, and it you want more later you can add more code.

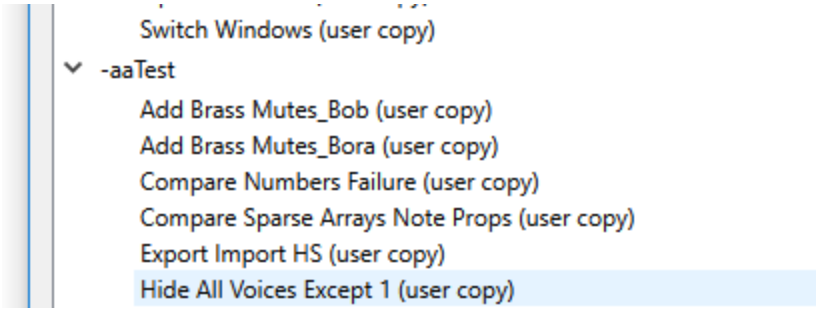
It is often useful to save plugins for things that are very hard or impossible to do quickly, or that you will do lots of times. In those cases you might write more code to handle more situations. Or just write code for the hard parts and do other things the normal way.

For now, let's just put together a plugin that will run this code. We will follow the methods described in http://www.bobzawalich.com/wp-content/uploads/2020/02/Write-your-own-simple-plugin-1.pdf, and I will give truncated explanations here.

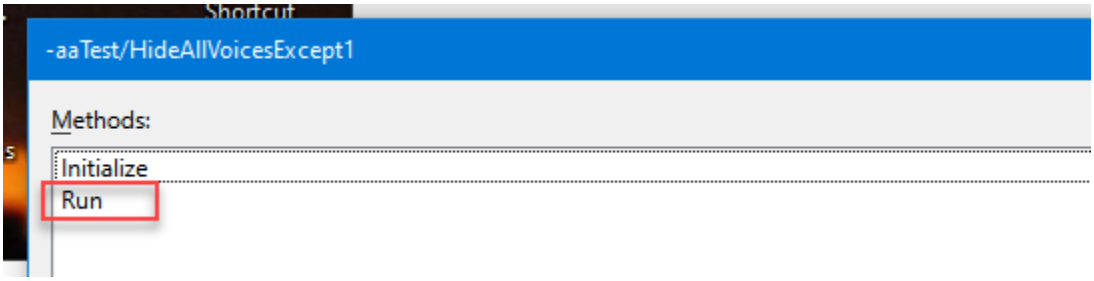
1. In Sibelius Ultimate, go to *File>Plug-ins>Edit Plugins*, and type *New*.
    - You will see a dialog. Give the plugin a name without spaces, which will become its file name, and a name with spaces, which will appear on the menu. Always make these the same, except for the spaces. I am calling this plugin *Hide All Voices Except 1*.
    - You need to give the plugin a *Category*, which is a subfolder name inside your user-plugins folder. I suggest making up a separate category for your personal plugins. I use *-aaTest*, which will sort to the top of the list of plugins when I look at the folders. *MyPlugins* or *aaaMyPlugins* are similar choices. Sibelius will create the category/folder if it is not there already.



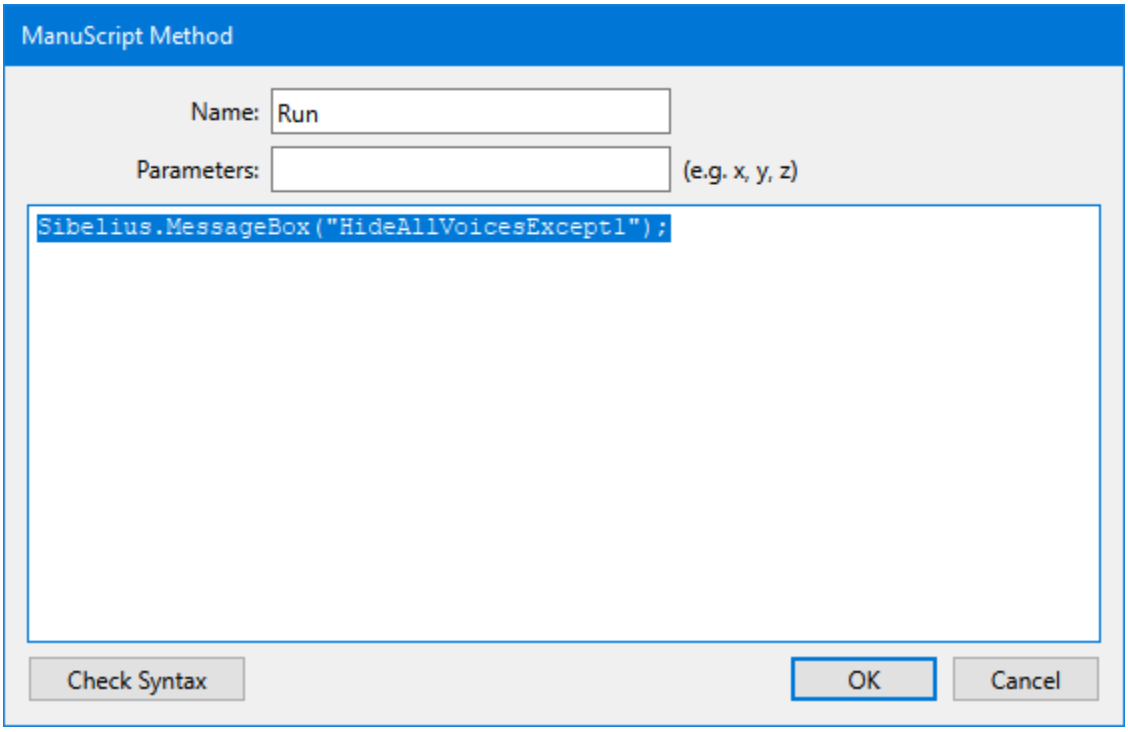
2. One big advantage of using *Edit> Plugin> New* is that it installs the plugin for you, and you can use or edit it immediately. Look at the plugin list in *Edit Plugin*, and in the folder *-aaTest* (or your name of choice) you will see your plugin. Double click it to edit it.



3. You will see a dialog that looks like this, and we are interested in the *Run* method. (A method is a block of code, like a function or subroutine in other languages).



4. Double click on *Run* and you will see this:

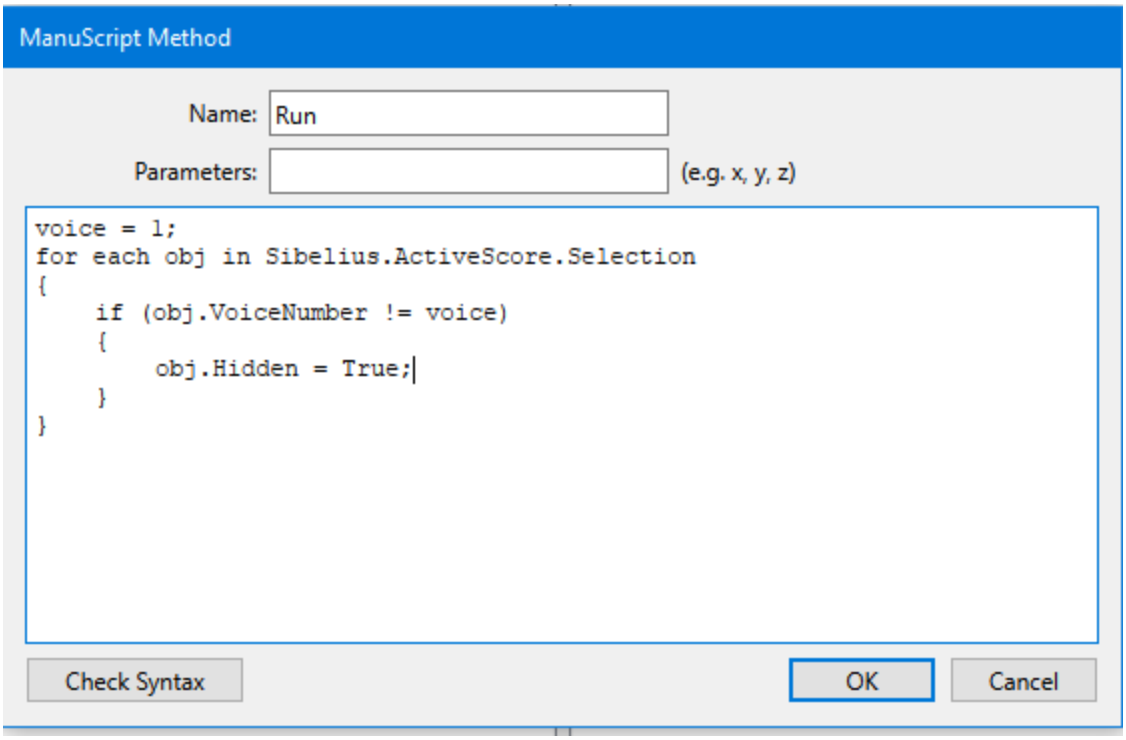


If you were to run the plugin as it is, it would put up a message box that displays the file name of the plugin. Not all that useful, but it is obvious that you ran it.

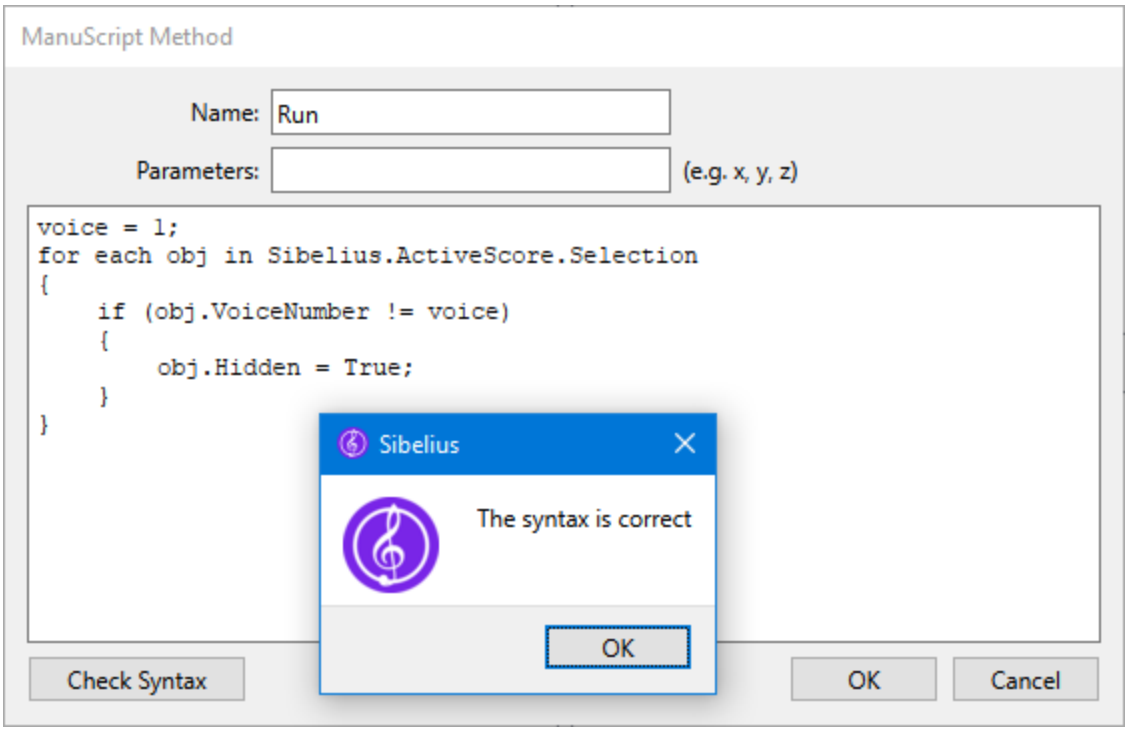
We want to replace this with our code, which is:

```
voice = 1;
for each obj in Sibelius.ActiveScore.Selection
{
   if (obj.VoiceNumber != voice)
   {
```

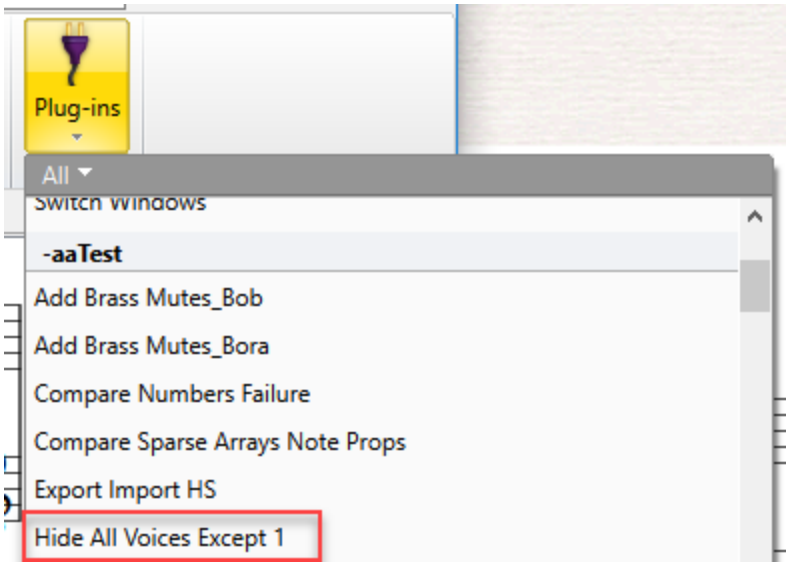
```
        obj.Hidden = True;
    }
}
```



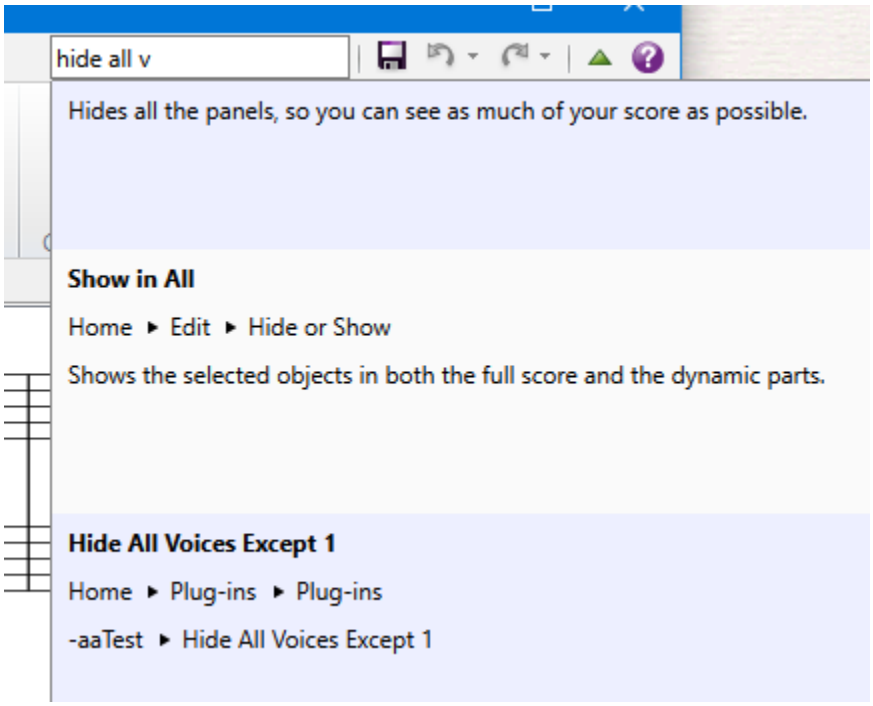
Once we edit the code, it is always wise to click the *Check Syntax* button to catch any typos.



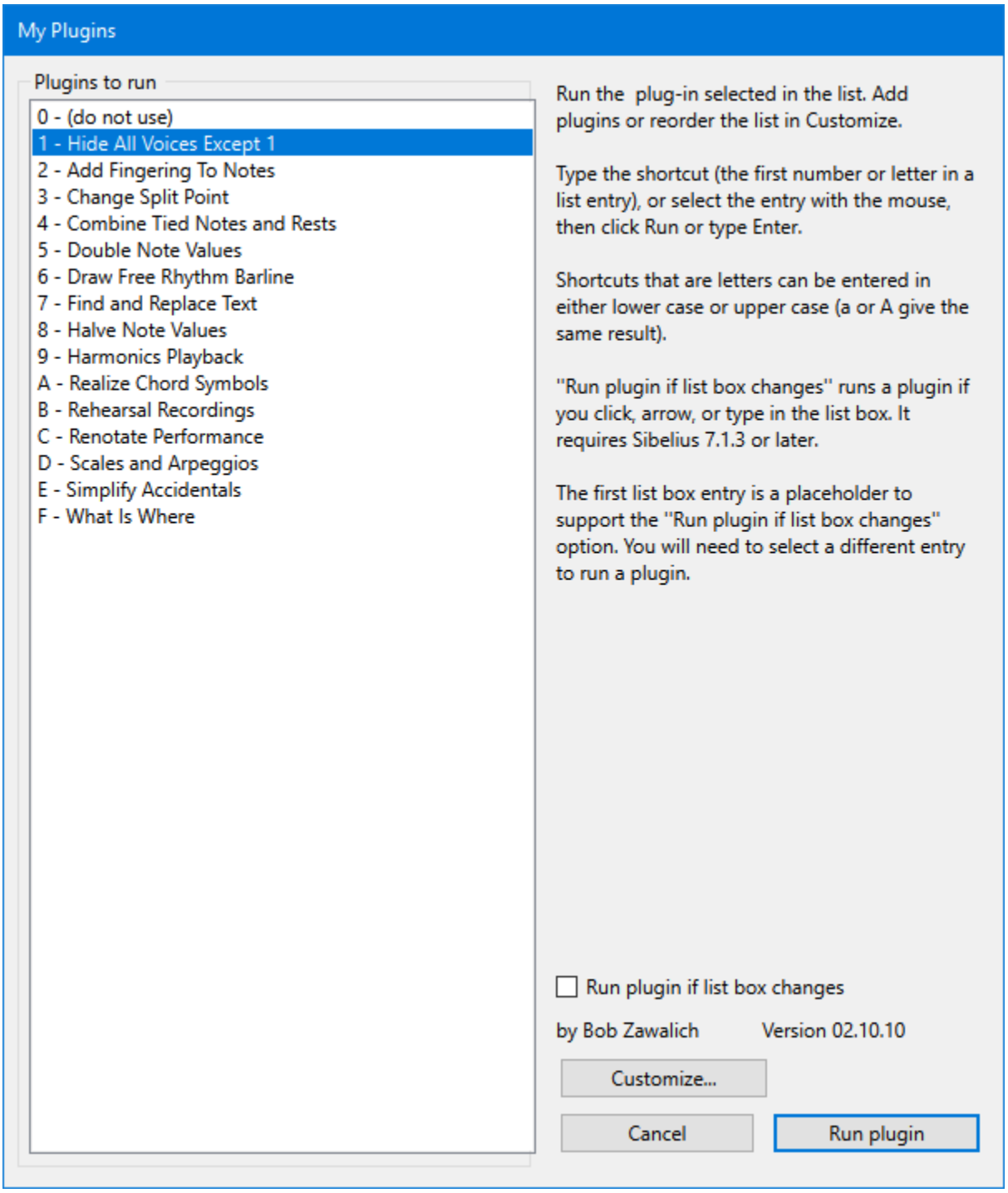
5. Now we want to run the plugin. There is a *Run* command in the plugin editor but you will want to be able to run it outside the editor so we might as well set that up.
    - You can run it from the Home menu, in *-aaTest*, which is where it resides by default



- You can type the name into the *Ribbon Search* box and run it from the list

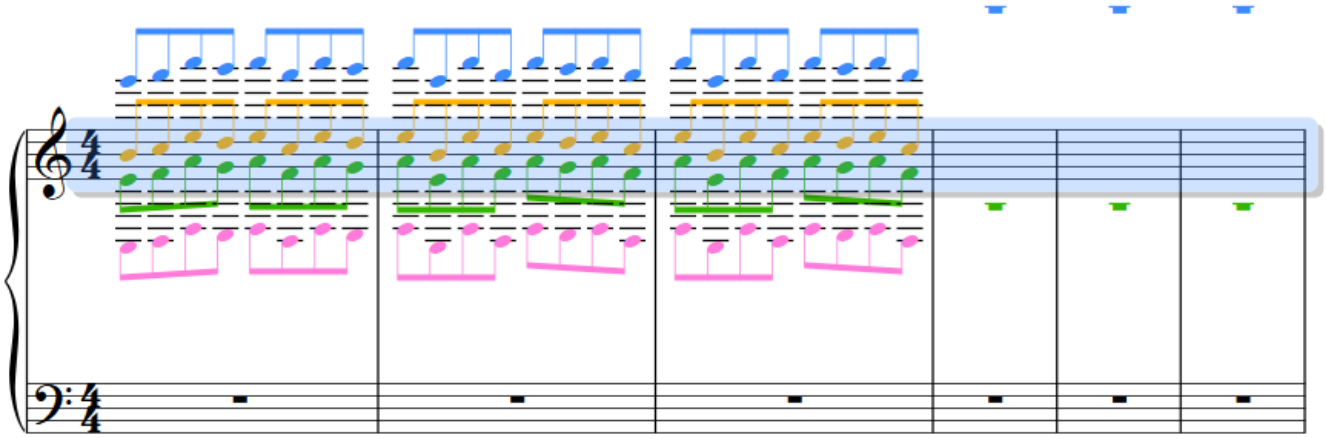


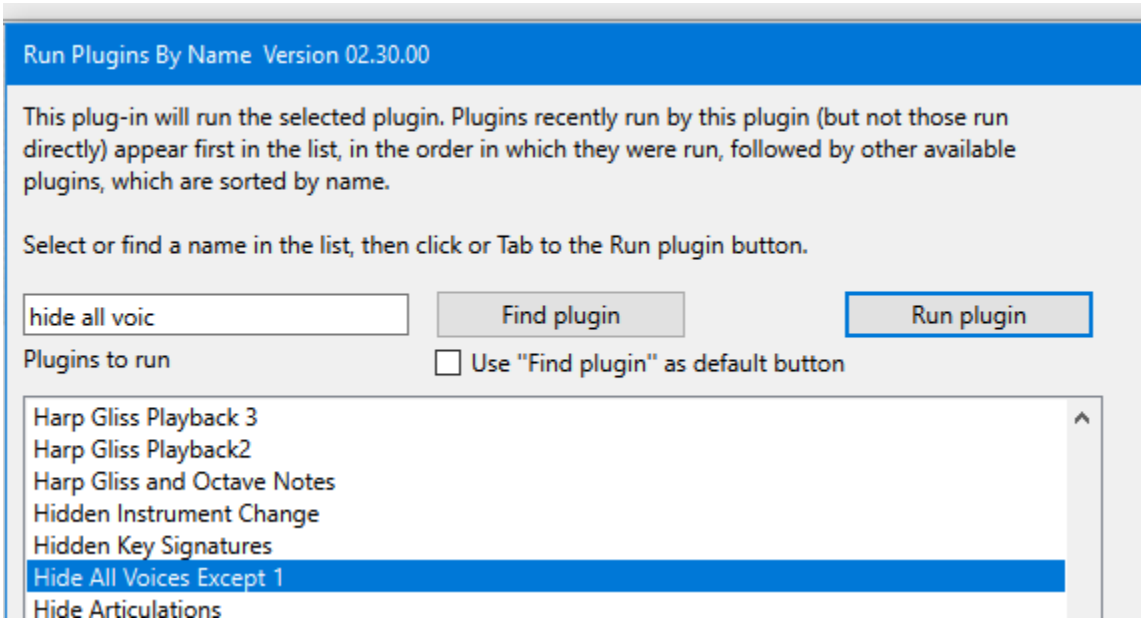
- You can go to *File>Preferences>Keyboard Shortcuts* and assign it a shortcut.
- You can use one of the plugins like *Run Plugin By Name* or *My Plugins*, and assign a shortcut to that plugin, and find it quickly in the plugin list. *My Plugins* is especially handy for setting up 2-key shortcuts to up to 35 plugins (with shortcuts 1 – 9 and a – z).



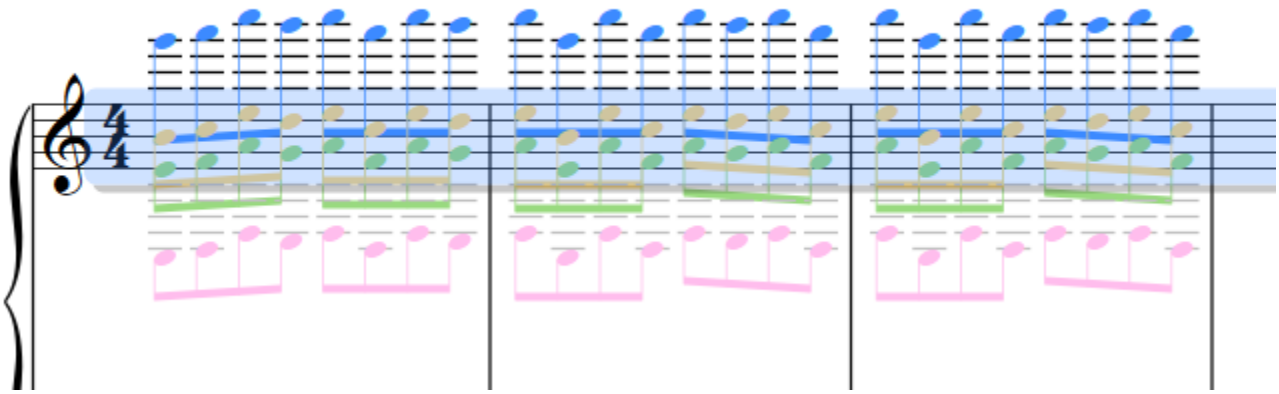
So, assuming you have a way to run your plugin, let's try it out. Here I have a score with notes in all 4 voices. I make a passage selection on some notes in multiple voices

Then I run the plugin (in my case using *Run Plugins By Name*)





And here I see that voice 1 is the only unhidden voice, which is of course what I asked for. This will look better it I turn off *View>Hidden Objects*. Note that the stem directions of voice 1 notes have reverted to what they would be if it were the only voice in the bars.



So. It did what I asked it to do. To handle the other voices, you could make copies of the files and edit them in a text editor, or just go back to the plugin editor and create 3 more plugins changing "*1*" to the voice you want in the file name and the name used in *Initialize*, and in the code in *Run*. (Shown as *XXXX* here).

```
voice = XXXX;  // XXXX is 1, 2, 3, or 4
for each obj in Sibelius.ActiveScore.Selection
{
   if (obj.VoiceNumber != voice)
   {
      obj.Hidden = True;
   }
}
```

There is more that could be done. I might want to mute the hidden voices, so after the line

```
      obj.Hidden = True;
```

I might use the routine

SetPlayedOnNthPass(*n, do play***)** Tells Sibelius whether or not the object should play back the *n*th time.

to turn off play on pass for pass 1 (and add additional lines if I need more passes turned off).

```
obj. SetPlayedOnNthPass(1, False);
```

Now the block of code is

```
voice = XXXX;  // XXXX is 1, 2, 3, or 4
for each obj in Sibelius.ActiveScore.Selection
{
   if (obj.VoiceNumber != voice)
   {
      obj.Hidden = True;
     obj. SetPlayedOnNthPass(1, False);
   }
}
```

If I were to use this, I might want to add code that would restore the voice I am not hiding to be visible and to play. Welcome to programming!

Give it a try and see what happens.