The Visual SDIF interface in PWGL

Mika Kuuskankare*

Sibelius Academy, Finland
mkuuskan@siba.fi

Abstract. In this paper we present a novel PWGL user-library, SDIF, that allows us to import SDIF encoded sound analysis information into PWGL. The library provides us with a visual box interface to the CNMAT/IRCAM SDIF Tools. The PWGL SDIF library builds on top of another PWGL library called SHELL enabling us to interact with the UNIX command-line. The SDIF library is still in early development and testing but it has already proven to be quite robust and functional. In this paper we introduce the current functionality of our library and discuss some concrete use cases and future development possibilities.

Keywords: SDIF, computer assisted composition, visual programming, visualization

1 Introduction

PWGL [9] is a Lisp-based music programming environment designed for the applications of computer assisted composition, music theory and analysis, software synthesis, and music notation. Currently, PWGL lacks tools that would allow us to straightforwardly use and manipulate sound analysis information in the visual patch. There are numerous free and commercial applications, such as AudioSculpt and SPEAR, that are able to perform sophisticated sound analysis and processing. The solution we propose here is to use third party applications to perform the analysis/processing and read the information into PWGL for further treatment. To this end we have implemented a new PWGL user-library, SDIF, that is able to import SDIF description files into our system.

SDIF (Sound Description Interchange Format, [12]) is a standard and extensible interchange format of sound descriptions jointly developed by IRCAM and CNMAT. SDIF consists of a large collection of spectral description types. Numerous programs currently support SDIF, among others Matlab [13], Max/MSP, PureData, jMax (through the FTM library), OpenMusic [2], Spear [6], AudioSculpt [4], ASAnnotation [3], and CLAM [1]. It has become de facto standard sound interchange format in the field of computer music and research.

In addition to the standalone applications several language bindings also exist, among others for C, Java, and Python. The Lisp-based SDIF interface is

* The work of Mika Kuuskankare has been supported by the Academy of Finland (SA137619). We would also like to thank CCRMA, Stanford University, for hosting the research.

9th International Symposium on Computer Music Modelling and Retrieval (CMMR 2012)
19-22 June 2012, Queen Mary University of London

provided by OpenMusic. In OpenMusic the SDIF interface [5] is implemented by calling the functions of a dynamic C-library directly using the Lisp Foreign language Interface (FLI). The SDIF data types as mirrored in the Lisp side using FLI data types.

Our interface, in contrast, provides direct access to the IRCAM/CNMAT SDIF Tools–a collection of command-line programs that help in reading, writing, and manipulating SDIF data files. Our library is implemented using a standard PWGL library called SHELL that allows us to interface with the UNIX command-line. SDIF operations are performed by the SDIF Tools and the results are read in PWGL through a pipe opened between PWGL and the shell. There are several advantages in this approach. First, it provides full access to the information distributed in SDIF format. Everything that can be done with the SDIF Tools can be done inside PWGL. Second, our box interface mirrors the functionality of the SDIF Tools. Thus, there's no need to learn a new box representation or naming conventions.

Eventually, the SDIF interface will allow us to develop many new PWGL applications. Our music notation program, ENP [7], in conjunction with the SDIF library, provides the users with a powerful toolkit, for example, for the applications of spectral composition. Our software synthesizer, PWGLSynth [10], would benefit from sophisticated sound analysis data difficult to obtain in any other way.

The PWGL SDIF user-library and the relating documentation can be downloaded from our project's web site at `http://www2.siba.fi/PWGL/downloads.html`

2 Overview of the SDIF Library

The PWGL SDIF library interfaces with the open-source CNMAT/IRCAM SDIF command-line tools. When the user loads our library for the first time, a PWGL specific version of the SDIF Tools is automatically compiled[1] and installed locally inside the SDIF library folder. Our version converts the output into a format understood by the Lisp reader, i.e, the results are in general always returned as a Lisp list.

The main utilities that we currently use are: `querysdif`, and `sdifextract`. `querysdif` displays a summary of the data in an SDIF file, and the ASCII header information. `sdifextract`, in turn, outputs the data of an SDIF file either as a whole or according to the options defined by the user.

The SDIF user-library is accompanied by a tutorial that gives several working examples dealing with compositional and sound synthesis applications. The documentation also provides the users with links to relevant study material that can be found either inside PWGL or on the internet.

Most of the functionality presented in the sections 2.1–2.3 the SDIF library inherits form the SHELL library. The PWGL SHELL library provides a collection

[1] The prerequisite for using the library is that the user has installed the free Apple Developer Tools.

of specialized box types that allow us to interface visually with the UNIX shell. Thus, it is possible to call virtually any shell program, to redirect and pipe commands, and to input the results back into the PWGL patch.

2.1 Managing Options

Command-line utilities usually allow users to define a variable number of different options. Managing and remembering all the possible options and the combinations thereof can be quite demanding. Our system provides the users with a visual browser (see Figure 1) for managing options. From the browser the user is able to select an appropriate option and study its documentation and even see concrete examples of its use. The SDIF library boxes behave like expert boxes in the sense that they provide relevant information for the users and assistance in their use.

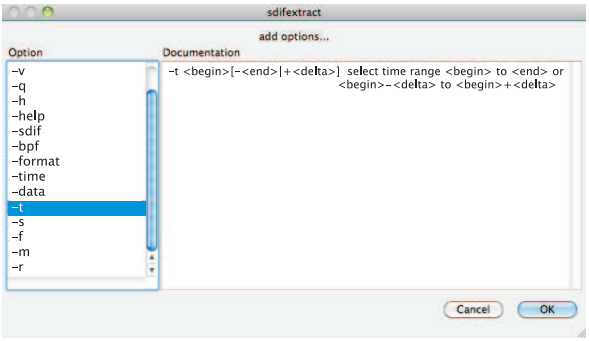


Fig. 1: *The SDIF library options dialog showing the options relevant to the sdifextract box.*

2.2 Error Handling

One important functionality of the SDIF library is its ability to gracefully handle errors that happened during the shell execution and to signal the user that something has gone wrong, e.g., a required file was not found. The UNIX error code is trapped by the boxes and displayed both visually and textually in the patch. The boxes can thus display pertinent information about their status and the success of the operation in question.

Figure 2 shows a box of the SDIF library in an erroneous state. A red warning sign is drawn over the box. When the user moves the mouse pointer over the box the relevant error message (corresponding to the UNIX error code) is displayed in a message area at the bottom of the patch window.

2.3 Argument Passing and Conversion

When interfacing with the SDIF Tools we need to be able to convert Lisp arguments into a form understood by the command-line utilities. The SDIF library

Fig. 2: *An SDIF library box showing that an error occurred while processing the user's request.*

provides a method with which the users can define translators from data written in Lisp to the data required by the utility.

Let us examine a brief example. Some of the SDIF tools accept time range as an option. For example, sdifextract allows us to define a time range for extracting only the relevant part of the audio analysis. This information is given using one of the following formats: `1.5-3.5` or `2.5+3.0`.

Normally, when passing this option from Lisp the users would have to write the data as a string, which makes its use inconvenient. In Lisp, the most convenient way of representing this information is a list of two numbers, e.g., '(1.0 2.5). The problem can be solved by adding a pre-processor to the SDIF box. In this case, we define a pre-processor for the given box type, `sdifextract`, and for a given option `-t` (as for time) as follows:

```
1 (add-pwgl-shell-box-pre-process
2 "sdifextract"
3 "-t"
4 #'(lambda(x)
5     (format () "~f-~f" (car x) (cadr x))))
```

The translator function (lines d-e) converts the incoming data. Now, the user can input the time range just by passing a list of two numbers.

3 SDIF Library Boxes

Here, we present the new PWGL box types (in addition to the two core boxes sdifextract and querysdif) that are unique to the SDIF library: (1) SDIF-selection box (2) SDIF-selection-spec box, (3) SDIF-range box, and (4) SDIF-type box.

3.1 SDIF-selection Box

Routinely there is a need to access only a subset of the information presented in an SDIF file. All SDIF tools accept an SDIF selection, which allows the users to specify relevant time ranges, frames or matrices that are accessed from the file. In our case this allows for a fast access to a part of the data contained by an SDIF file. Furthermore, the files do not need to be processed inside PWGL using Lisp; the processing is done by the SDIF library itself. This reduces the workload in PWGL and also allows us to remain compatible with the SDIF Tools collection as the implementation is kept outside our system. Incidentally, our flexible box

design scheme should allow us to keep up with the potential changes in the SDIF selection syntax.

The syntax of the selection is specified in [11] and is as follows:

```
[filename]::[#stream][:frame]
[/matrix][.column][_row][@time]
```

For convenience we have defined a special PWGL box (see Figure 3) that allows us to define the SDIF selection using Lisp objects. The box accepts a variable number of arguments in an arbitrary order. It sorts its arguments and outputs a syntactically valid SDIF selection object that can be passed as an argument to relevant SDIF library boxes.

Fig. 3: *An SDIF-selection box with two options: frame and time.*

The box with the options shown in Figure 3 translates to the following piece of SDIF selection code:

```
[...]/pf-w.pd-hit-fnl.fft.sdif::1HRM@0.0-2.0
```

3.2 SDIF-selection-spec Box

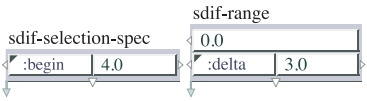
SDIF-selection-spec (Figure 4a) box converts lisp representations to various other formats required by the SDIF Tools. This box can be used with the SDIF-selection box to define ranges, or comma separated lists when needed. In SDIF, numeric values can be represented either as lists (e.g., selecting only the columns 1 and 2) or ranges (e.g., selecting the rows 1-50). The box also allows us to specify incomplete ranges as per SDIF specification (e.g., where the lower or upper value is replaced by the respective minimum or maximum value). The box shown in Figure 4a translates to an (incomplete) SDIF range specification `4.0-` (i.e., beginning from 4.0 seconds until the end).

3.3 SDIF-range Box

SDIF-range (Figure 4b) box converts lisp representations to the SDIF time range format. This box accepts the input in several different formats as per SDIF specification. The box shown in Figure 4b translates to a range of `0.0+3.0`.

3.4 SDIF-type Box

The SDIF Type box provides the users with a browsable dialog containing all the SDIF types and any relevant information about them. The box can be connected to other SDIF library boxes that require SDIF types as parameters. Multiple selection is also allowed.



(a) (b)

Fig. 4: *The SDIF library utility boxes: (a) the SDIF-selection-spec box, and (b) the SDIF-range box.*

4 Some Use Cases

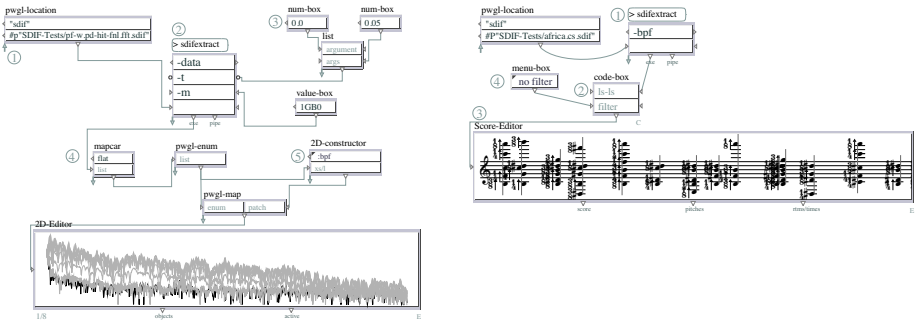
In this section we give two use cases of the SDIF library. In our first example, we visualize FFT data stored in an SDIF file. The second example deals with a compositional application where we read into PWGL chord sequence analysis information, prepared with the help of AudioSculpt, and convert it into a musical score.

4.1 Data Visualization

Figure 5a shows a patch where we read FFT information stored in an SDIF file and visualize it using our 2D-Editor [8]. The SDIF file pathname is given in (1) as an argument to the `sdifextract` box in (2) which, besides the mandatory pathname argument, has three options. The `-data` option instructs the box to return only the data without times. The `-t` option gets as an argument a time range, i.e., we read in only the frames between 0 and 5 milliseconds. The `-m` option allows us to select which matrix to extract. In this case, we are interested in matrices of the type `1GB0` which contain the FFT data. As the 2D-Editors can display information in several independent layers we use the 2D-constructor box (5) to create individual breakpoint-functions of each of the FFT-frames. The result is shown in the 2D-Editor at the bottom of the patch.

4.2 Score Interface

Figure 5b demonstrates how to manipulate SDIF data in PWGL to generate a sequence of chords. The analysis data is read in (1) using the `-bpf` option as we need the frame times as well as the frequency data. In (2) we convert in the code-box (the code is not shown here) the data to a list of chords and the result is given to the Score-Editor box (3). Here, the user can apply a filter (4) to get all notes, only notes that have velocity values below the average velocity of the chord, or only notes that have velocity values that are above average. Note also, that ENP allows us to represent micro tones using a dynamic resolution. Here, an eighth-tone resolution is defined by the user.



(a) (b)

Fig. 5: *The examples demonstrating the usage of PWGL SDIF user-library: (a) visualizing FFT information stored in an SDIF file, and (b) manipulating FFT data to generate symbolic music notation.*

5 Future Development

Currently, it is only possible to read into PWGL analysis information prepared outside our system. Next, we plan to extent the current interface scheme so that it supports making the analysis itself using the patch language. The extension would provide us with an access to SuperVP or pm2 (two sound processing tools developed by the Analysis/Synthesis team of IRCAM) or other similar analysis tools.[2] Furthermore, it should eventually be possible to write SDIF files directly from PWGL. This would make it possible to process the SDIF information in a patch and save the processed material again in SDIF format.

6 Conclusions

In this paper we present a new PWGL user-library that allows us to access and manipulate different kinds of sound analysis information visually by interfacing with the SDIF file format. The library does not use foreign language bindings, but instead interfaces with the UNIX command-line tools. We describe the current state of the library, present the basic concepts and the functionality of the visual box interface. We also cover some of the potential applications by demonstrating how to visualize SDIF data and convert it to high-level musical score representation.

[2] The possibility of using AudioSculpt kernels would naturally be available only for users that have a licensed copy of the software.

References

1. Amatriain, X., Arumí, P.: Developing cross-platform audio and music applications with the clam framework. In: Proceedings of the International Computer Music Conference. pp. 403–410 (2005)
2. Assayag, G., Rueda, C., Laurson, M., Agon, C., Delerue, O.: Computer Assisted Composition at IRCAM: From PatchWork to OpenMusic. Computer Music Journal 23(3), 59–72 (Fall 1999)
3. Bogaards, N., Yeh, C., Burred, J.J.: Introducing asannotation: a tool for sound analysis and annotation. In: Proceedings of International Computer Music Conference (2008)
4. Bogaards, N., Röbel, A., Rodet, X.: Sound analysis and processing with audiosculpt 2. In: Proceedings of International Computer Music Conference. Miami, USA (2004)
5. Bresson, J.: Sound processing in openmusic. In: Proceedings of the 9th International Conference on Digital Audio Effects – DAFx-06. pp. 325–330. Montréal, Canada (2006)
6. Klingbeil, M.: Software for spectral analysis, editing, and synthesis. In: Proceedings of the International Computer Music Conference (2005)
7. Kuuskankare, M., Laurson, M.: Expressive Notation Package. Computer Music Journal 30(4), 67–79 (2006)
8. Laurson, M., Kuuskankare, M.: PWGL Editors: 2D-Editor as a Case Study. In: Sound and Music Computing '04. Paris, France (2004)
9. Laurson, M., Kuuskankare, M., Norilo, V.: An Overview of PWGL, a Visual Programming Environment for Music. Computer Music Journal 33(1), 19–31 (2009)
10. Laurson, M., Norilo, V., Kuuskankare, M.: PWGLSynth: A Visual Synthesis Language for Virtual Instrument Design and Control. Computer Music Journal 29(3), 29–41 (Fall 2005)
11. Schwarz, D., Wright, M.: Extensions and applications of the sdif sound description interchange format. In: In Proceedings of the International Computer Music Conference. pp. 481–484 (2000)
12. Wright, M., Chaudhary, A., Freed, A., Wessel, D., Rodet, X., Virolle, D., Woehrmann, R., Serra, X.: New applications of the sound description interchange format. In: International Computer Music Conference. pp. 276–279. International Computer Music Association, Ann Arbor, Michigan (1998), `http://cnmat.berkeley.edu/publications/new_applications_sound_description_interchange_format`
13. Wright, M., III, J.O.S.: Open-source matlab tools for interpolation of sdif sinusoidal synthesis parameters. In: Proceedings of International Computer Music Conference. pp. 632–635 (2005)