Distributing Music Scores to Mobile Platforms and to the Internet using INScore

Dominique Fober, Guillaume Gouilloux, Yann Orlarey, Stéphane Letz

Distributing Music Scores to Mobile Platforms and to the Internet using INScore

D. Fober **G. Gouilloux** **Y. Orlarey** **S. Letz**
GRAME
Centre national de création musicale
Lyon - Fr
fober@grame.fr

ABSTRACT

Music notation is facing new musical forms such as electronic and/or interactive music, live coding, hybridizations with dance, design, multimedia. It is also facing the migration of musical instruments to gestural and mobile platforms, which poses the question of new scores usages on devices that mostly lack the necessary graphic space to display the music in a traditional setting and approach. Music scores distributed and shared on the Internet start also to be the support of innovative musical practices, which raises other issues, notably regarding dynamic and collaborative music scores. This paper introduces some of the perspectives opened by the migration of music scores to mobile platforms and to the Internet and it presents the approach adopted with INScore, an environment for the design of augmented, interactive music scores.

1. INTRODUCTION

When you search *"music score app"* on the Internet, you'll likely get more than 39,000,000 matching pages when associated to *"android"*, more than 12,000,000 when associated to *"iOS"* and over 29,000,000 with *"iPad"* as keyword. Adding *"Web"* or *"Internet"* to the query results in an explosion of matching pages, while support for historical operating systems tends to be bygone (figure **??**). These figures indicate clearly a significant evolution and a clear migration of the support for musical scores to mobile devices but above all, to the Internet.

From a technical viewpoint, this change represents a move from one operating system [OS] to another. Web browsers may be viewed as a kind of OS on top of an abstract machine: they are integrating step by step all the services of an OS, up to audio services with the recent Web Audio API [**?**]. But actually the change is far more than this simple move:

- mobile platforms have adopted a fundamentaly different approach from user point of view: no traditionnal input device (keyboard, mouse), embedded sensors that may be used as controlers, a reduced graphic space especially for smartphones, a step back on multitask aspects but an wave of services composition and integration.
- web applications differ also due to their natural way to agglutinate distributed resources and to share content between several users. Theoretically, they can be deployed on all the platform previously mentioned.

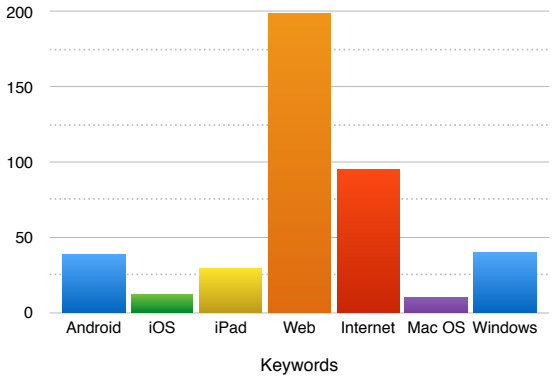

Figure 1. Results of a search using Google with *"music score app"* associated to different keywords. The numbers are in million hits.

The approach adopted by almost all music notation applications available on mobile platforms is rather classical: you can find a plethora of music score readers and players, based on the common music notation, more or less sophisticated. Music score edition is also supported by these applications but they have to re-think the user interface due to the lack of input device: handwritten recognition is one of the explored solutions [1] [2]. On smartphones, the screen size limitation is not really handled, apart with messages that inform the user that the application may not be fully functional. The more innovative approaches generally come from artistic uses [**?**].

On the web side, you can find online music notation editors, online score sharing systems [3], or JIT compilation services like those proposed for years by the GUIDO Engine [**?**]. More recently, music notation services have been made available to developers and users under the form of

[1] NotateMe http://www.neuratron.com/notateme.html
[2] StaffPad http://www.staffpad.net/
[3] MuseScore https://musescore.org/en/handbook/share-scores-online-0

a RESTFUL web service [**?**]. Solutions for score layout and redering can also be embedded in a web page: this is the case for the GUIDO Engine [**?**] that is now available as a Javascript library. Applications for music practising are now moving to the Internet (e.g. Weezic [4]) but based on already existing strategies [**?**] [**?**]. Distributed and collaborative scores are appearing like the Flat music score editor [5] and also tools for network improvisation [**?**].

Whether running on the web or on mobile platforms, the approach to music notation adopted by almost applications looks quite classical. Innovation generally comes from artistic approaches and are based on specific tools. However, the context of mobile platforms or of the Internet could lead to new and original uses, and we think that in this regard, an adequate support is missing from tools for music notation.

This paper proposes several use cases that are specific to the context described above. These use cases may be implemented using INScore, an environment for the design of augmented interactive music scores that has been extended to support distributed scores and collaborative design, and that runs on all the major platforms (MacOS, Linux, Windows, Android and iOS). The paper starts with a brief reminder of the INScore environment. Next it presents the network extensions. Various use cases are then considered and a concrete realization of a dynamic score published over Internet is presented.

2. INSCORE

INScore is an environment for the design of augmented, interactive music score [**?**] that is entirely controled by an Open Sound Control [OSC] API [**?**]. It supports arbitrary graphic resources (symbolic music notation, text, images, vectorial graphics...) and displays the time relationships of the score components by the way of a simple synchronization mechanism. INScore supports performance representation, viewed as audio or gestural signal, as well as interaction process representation also viewed as signals [**?**]. It includes an event based interaction mechanism [**?**] that provides a simple and homogeneous way to describe interactions in the graphic or the temporal space.

INScore input language is a textual version of OSC messages extended to support variables, extended OSC adresses and Javascript sections. A Javascript engine is embedded in each score and may be remotely triggered via OSC messages.

The script below shows the example of a rectangle synchronized on a symbolic score (described using the Guido Music Notation format [GMN] [**?**]) i.e. its graphic position is computed frm it's time location. The result is illustrated in figure **??**.

[4] `http://www.weezic.com/`
[5] `https://flat.io/`

EXAMPLE 1:

```
/ITL/scene/score set gmn '[g e c a f]';
/ITL/scene/rect set rect 0.05 0.3;
/ITL/scene/rect color 0 0 240 120;
/ITL/scene/sync rect score;
/ITL/scene/rect date 3 4;
```

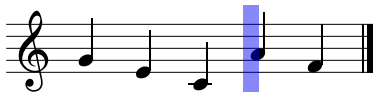

Figure 2. A rectangle synchronized on a score.

INScore has been used in many artistic projects, the last one being an implementation of Earl Brown's December Variation by Richard Hoadley [**?**].

3. INSCORE WEB SUPPORT

INScore has been extended to support aggregation of distributed resources over Internet, as well as publication of a score via the HTTP and the WebSocket protocols.

3.1 Distributed score components

Most of the components of a score can be specified in a litteral way or using a file. The example below will produce the same object, provided that the '`score.gmn`' file contains the `[g e c a f]` code.

EXAMPLE 2:

```
/ITL/scene/score set gmn '[g e c a f]';
  is similar to
/ITL/scene/score set gmnf 'score.gmn';
```

All the file based resources can be specified as a simple file path using absolute or relative path, or as an HTTP url. When using the relative path form, a file absolute path is built using the score current path, that may be set to arbitrary location using the `rootPath` message. This current path can be also be set to an arbitrary HTTP url, so that further use of a relative path will also result in an url.

The example below refers the same '`score.gmn`' file on `host.domain.org`.

EXAMPLE 3:

```
/ITL/scene/score set gmnf
    'http:://host.domain.org/score.gmn';
  is equivalent to
/ITL/scene rootPath
    'http:://host.domain.org/';
/ITL/scene/score set gmnf 'score.gmn';
```

This mechanism allows to mix local and remote resources in the same music score, but also to express local and remote scores in a similar way, using just a `rootPath` change.

3.2 HTTP and WebSocket components

A music score can be published on the Internet using the HTTP or the WebSocket protocols. Specific components can be embedded in a music score in order to make this score available to remote clients:

- an HTTP server, which INScore type is `httpd` and that takes a listening port number as argument,
- a WebSocket server, which type is `websocket` and that takes a listening port number and a maximum rate for clients notification as arguments.

The WebSocket server allows bi-directional communication between the server and the client. It sends notifications of score changes each time the graphic appearance of the score is modified, provided that the notification rate is lower than the maximum rate set at server creation.

The example below creates an HTTP server that responds on the port 8000 and a WebSocket server that responds on the port 8100 and sends notifications at a maximum rate of 200 ms.

EXAMPLE 4:
```
/ITL/scene/http set httpd 8000;
/ITL/scene/ws set websocket 8100 200;
```

The communication scheme between a client and an INScore server relies on a reduced set of messages. These messages are protocol independent and can be equally supported over HTTP or WebSocket. Table **??** gives an overview of the client server communication scheme:

- the `get` message requests an image of the score. It is similar to an `export` message adressed to INScore, which result is sent over HTTP or WebSocket.
- the `version` message requests the current version of the score. The server answers with an integer value that is increased each time the score is modified. This message is intended to allow clients to keep an up-to-date image of the score. Note that the WebSocket server automatically sends changes notifications with versionning information.
- the `post` message is intended to send an INScore script to the server. The server answers with a status message which is between `OK` or `ERROR`. In case of error, details on the failure reason are provided. In case of success, the score may be modified and its current version number is increased.
- the `click` message is intended to allow remote mouse interaction with the score. The associated data should be a position in an image previously retrieved with a `get` message.

3.3 Messages forwarding

Message forwarding is another mechanism provided to distribute scores over a network. It is applied at application and score levels. It consists in a list of destination

Request	Data	Answer	Side effect
`get`	*none*	an image	*none*
`version`	*none*	version num	*none*
`post`	INScore script	status	new score
`click`	x, y position	*none*	new score

Table 1. INScore server Web API.

hosts specified using a host name or an IP number, and suffixed with a port number. All the OSC messages may be forwarded to the indicated hosts on the corresponding port number, provided they are not filtered out (figure **??**). The filtering strategy is based on OSC adresses and/or on INScore methods (i.e. messages adressing specific objects attributes).

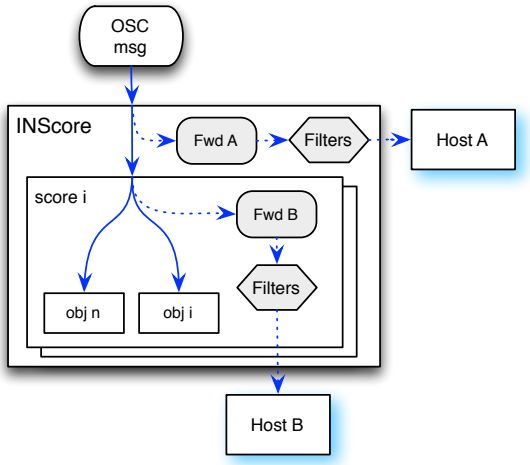

Figure 3. Message forwarding mechanism.

The next example installs a forwarding mechanism at application level: all the incoming messages may be forwarded to a host specified by IP number on the UDP port 7000. Next the filter is configured so that `clock` and `date` messages will not be forwarded.

EXAMPLE 5:
```
/ITL forward 192.168.1.27:7000;
/ITL/filter reject clock date;
```

4. USE CASES

4.1 Groupware technologies

Groupware technologies as described in [**?**] may be easily deployed using INScore and the forwarding mechanism. Let's say that we have a teacher score on a station T and students on 3 stations $S1$, $S2$, $S3$.

The settings illustrated in figure **??** can be implemented with the following messages:

A) on T : `/ITL forward S1 S2 S3;`

B) on T : `/ITL forward S1;`
on S1 : `/ITL forward T;`

C) on T : `/ITL forward S1 S2 S3;`
on S1 : `/ITL forward T;`

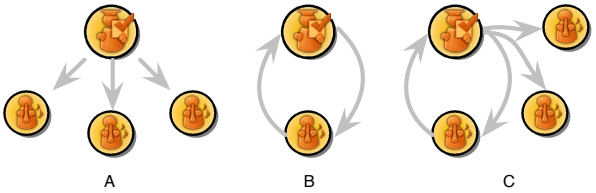

Figure 4. Use cases in a pedagogic setting: A)the teacher score is published to the students, B) the teacher and the student interact with the same score, C) the setting is similar to B) but the score is published to the other students that can look at the interaction.

4.2 Collaborative score design

Collaborative score design could be implemented with any number of participants, i.e. all the participants can interact with a score that is available to all the others, also in read/write mode. We assume that one station is the central point of messages distribution, then the forwarding scheme illustrated in figure **??** is describe below:

EXAMPLE 6:
```
on A: /ITL forward B C;
on B: /ITL forward A;
on C: /ITL forward A;
```

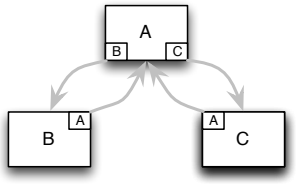

Figure 5. Collaborative score design.

Note that the forwarding scheme could be setup from the same computer using INScore extended OSC addresses (e.g. `B:/ITL/forward A`)

Note also that the forwarding mechanism prevents messages to be forwarded to the sender and thus, avoids direct loops (but not indirect loops e.g. A $\rightarrow$ B $\rightarrow$ C $\rightarrow$ A).

4.3 Shared score over Internet

Although the forwarding scheme above is basically intended to run on a local network, it could be implemented over the Internet as well, but since the underlying communication protocol is UDP, it may face significant packets losses, depending on the network conditions.

A secure solution to collaborative design may use the HTTP or WebSocket servers. The example below implements a score that displays the local score and includes a remote score as illustrated in figure **??**.

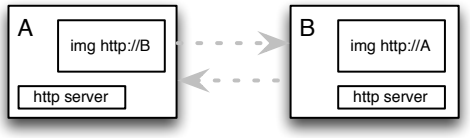

Figure 6. Scores shared over the Internet.

EXAMPLE 7:
```
On each station:
    /ITL/scene/http set httpd 8000;
    /ITL/scene/remote set img
              'http://remote.address';
```

Note that using a `websocket` object instead of `httpd` could make the remote view refreshment transparent. To do so, the `ws://` protocol has to be implemented for file based resources. That's one of the future directions.

4.4 Audience score based interaction

The INScore internet protocols support UI interactions and notably, relay the user clicks or touch screen interactions to the server. It is thus easy to imagine a concert setting where the music score is published (e.g. using the `websocket` server) and where the audience could get the score on a mobile phone and interact with it in real-time using the event based interaction mechanism of INScore, modifying the course of the music piece.

4.5 Flux Aeterna

Flux Æterna has been composed by Vincent Carinola in 2014. The piece has been designed for the Internet[6]. It comes under the form of an endless audio stream. The listening conditions are similar to those of a web radio but here, the listener can influence the future of the piece by providing its own sound files.

A dynamic score of the piece has been designed using INScore. The piece is using Max/MSP that sends modules and events information to INScore in real-time via OSC. This information is converted into a graphic information that reflects the piece structure (figure **??**) using the embedded Javascript engine.

The score has been initially designed for a local display, in the context of an exhibition. Adding a simple `httpd` or `websocket` object to the score allows to make it public over Internet, as illustrated with the HTTP example below:

EXAMPLE 8:
```
/ITL/scene/server set httpd 8000;
  makes the score available at
http://thehost.thedomain.org:8000
```

In addition, the score may be distributed in real-time to any INScore viewer connected to the local network using the forwarding mechanism mentionned in section **??**. The script below forwards the messages to any INScoreViewer

[6] `http://vr.carinola.free.fr/fluxaeterna/`

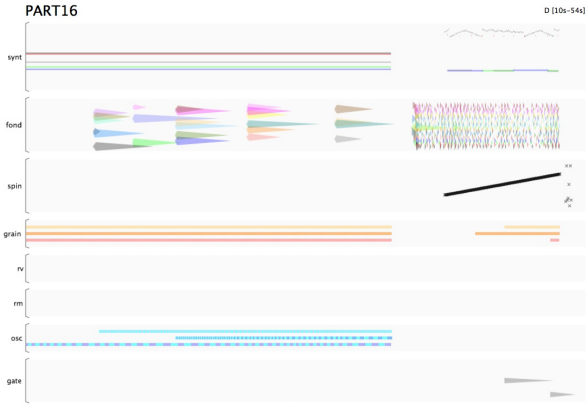

Figure 7. One page of Flux Æterna.

running on a device connected to the local network. Messages addressed to the Javascript engine are filtered in order to only forward the result of their evaluation.

EXAMPLE 9:

```
/ITL forward 192.168.1.255;
/ITL/filter reject '/ITL/scene/javascript';
```

5. CONCLUSIONS

Applications for music notation are moving to mobile platforms and to the web, following the general stream of computing migration. Most of these applications are reproducing the existing approaches to music notation although their deployment on the web and/or mobile platforms could take advantage of the technological context to create innovative uses. Actually, innovation exists but it is restricted to specific applications, mostly designed in artistic projects. With its network extensions and its support for Android and iOS, INScore provides a set of solutions for distributed score design and interaction. The approach tends to make network support as transparent as possible in a score description. Future extensions should make remote resources available using the `WebSocket` protocol, which should make remote files refreshment transparent and allow additional use cases in the domain of shared and collaborative score design.

Acknowledgments

INScore research and development has been funded by the French National Research Agency [ANR].
INScore is an open source project hosted on SourceForge (`http://inscore.sf.net`)