MAXSCORE: MUSIC NOTATION IN MAX/MSP

Nick Didkovsky
Rockefeller University, New York
www.algomusic.com
didkovn@mail.rockefeller.edu

Georg Hajdu
Hochschule für Musik und Theater Hamburg
www.georghajdu.de
georg.hajdu@hfmt-hamburg.de

ABSTRACT

This paper presents MaxScore, a Max object that provides standard western music notation in Max/MSP (Puckette, Zicarelli). MaxScore supports a rich set of Max messages that allows the user to populate a score with notes, query note properties, transform notes, play a score through Max/MSP via a well-defined instrument interface, and export a score in a variety of popular notation formats including MusicXML and LilyPond (Nienhuys). Transcription of Max-generated music is provided as well as note entry and editing by mouse or under program control. MaxScore supports user-defined plug-ins written in Java. We also present two applications, which utilize MaxScore: one that generates compositions in real-time, and another, which performs spectral transcription. MaxScore is written in Java Music Specification Language (Didkovsky, Burk) but requires no Java programming to use.

1. INTRODUCTION

Max/MSP is a widely used graphical environment for creating computer music and multimedia works using a paradigm of graphical modules and connections. Missing from Max is the capability to utilize standard western music notation directly within the Max environment. Java Music Specification Language is a Java API for music composition and interactive performance, and includes a notation package. While Max's Java API can be used to open JMSL's "ScoreFrame" notation editor (Didkovsky, Crawford), JMSL is not designed to receive Max messages or be further controlled by Max.
MaxScore is a Max object written in JMSL, which provides music notation directly within the Max environment. It supports a rich set of Max messages to create a score, populate it with notes in a variety of ways including: a) mouse entry, b) programmatically using its "addnote" message, and c) by using its transcriber. MaxScore also provides messages to transform existing musical material and to play back through Max so the score can control MSP patches. The MaxScore is rendered in its own Max LCD window (canvas), or can be embedded directly into the Max patcher (bcanvas). LCD was chosen in part because its set of drawing messages and event handling mapped efficiently from the commands used by JMSL's score canvas. Traditional music notation provides the Max composer with a rich set of possibilities for creating new work. It provides a bridge to a legacy of traditional musical practice and as such can provide performance materials in a format that is immediately understood by an enormous population of musicians playing traditional instruments. We also believe that there are times when the composer may find it more appropriate or more comfortable to specify compositional behavior in traditional notation. The availability of music notation significantly augments the range of works possible with Max.

2. MAXSCORE FEATURES

Notes can be added to a MaxScore with the addNote message, which takes as arguments the duration, pitch, amplitude, and hold time of a note (where duration of 1 is a quarter note, 0.5 is an eighth note, 0.3333 is an 8th triplet, etc). Of course the user could alternatively represent durations as whole number ratios by performing the division at the Max level and sending the floating-point quotient to addNote's duration parameter. JMSL's new auto-beaming feature beams notes to the beat of the current time signature as they are entered.

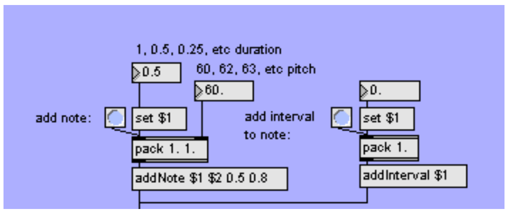


Figure 1. MaxScore's addNote message populates the score under program control.

Mouse entry of notes is also possible. By right/ctrl-clicking on a staff a note is entered. The user can select notes for copying, deletion, or alteration by dragging the mouse. Selection can also be done programmatically using the selectNote message which takes as arguments the measure number, staff number, track number, and note number of the note to be selected.
JMSL's transcriber (Didkovsky 2004) is also available to MaxScore, and can transcribe arbitrarily generated musical events. New to JMSL is a LilyPond exporter, which allows users to have their scores typeset by LilyPond music engraving software.

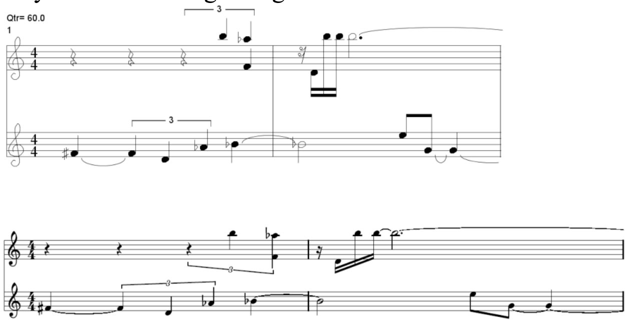
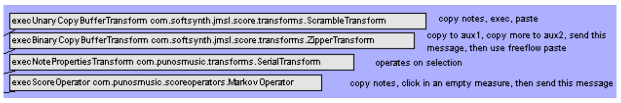


Figure 2. Two measures of Max-generated music transcribed by MaxScore are shown at the top half of the figure. In the bottom half of the figure the same two measures are shown, typeset by LilyPond after being exported by MaxScore.

JMSL's Transforms are available to MaxScore as well. Unary Copy Buffer Transforms operate on notes stored in the Copy Buffer. The results of the transform are pasted into the score by the user. Binary Copy Buffer Transforms operate on two distinct copy buffers and again, results are pasted into the score by the user. Note Properties Transforms operate in-place on selected notes (such as transposition or adding expressive marks). Finally, Score Operators perform arbitrary operations on a Score. As is the case with standard JMSL, MaxScore has the capability to scan the Java classpath for plug-ins that are of these types, and can populate a selection list of valid plug-ins. Selecting one from the list executes the plug-in.



```
execUnaryCopyBufferTransform com.softsynth.jmsl.score.transforms.ScrambleTransform      copy notes, exec, paste
execBinaryCopyBufferTransform com.softsynth.jmsl.score.transforms.ZipperTransform      copy to aux1, copy more to aux2, send this
                                                                                         message, then use freeflow paste
execNotePropertiesTransform com.punosmusic.transforms.SerialTransform      operates on selection
execScoreOperator com.punosmusic.scoreoperators.MarkovOperator      copy notes, click in an empty measure, then send this message
```

Figure 3. MaxScore supports JMSL plug-ins

Playback through Max/MSP is made possible by the new MaxScoreInstrument object. When a new MaxScore is created, each staff is assigned a new instance of a MaxScoreInstrument. When played back, these instruments pipe their performance data out MaxScore's second outlet, where it can be sent to MSP patches. The data is in the form of an ordered list, where the values are: instrument index, maxclock timestamp, pitch, amp, holdtime, "event flag", followed by additional user defined parameters unique to the instrument. The event flag indicates whether the note is tied in or not. Tied notes can be used to update timbral parameters during sustained sounding of an instrument. The MaxScore help file includes a working example that users may modify.
Information about notes in a MaxScore is available through a variety of querying messages. Some of these messages report terse information such as getNotePosition which dumps the measure number, staff number, track number, and note index of selected note, while other messages like the more general getNoteInfo dumps a rich set of properties in XML format which the Max user may parse for details.

3. MAXSCORE, SOME TECHNICAL DETAILS

MaxScore is written in Java and uses Java Music Specification Language's "score" package to provide both notation and musical scheduling. No Java programming is required of the Max user, as Max messages provide the user with control over the score and its contents. MaxScore extends MaxObject and so, can be imported into Max with the mxj message.
MaxScore renders to a Max LCD object. Our task was to provide JMSL with a graphics environment whose drawing commands would ultimately be sent out to Max. Since JMSL renders a Score to any Java class that implements the ScoreCanvas interface, we implemented a MaxScoreCanvas, which holds a reference to a Java Graphics subclass we called MaxScoreGraphics. When MaxScore creates a new JMSL Score, MaxScore hands the score a new MaxScoreCanvas as well as a reference to itself. MaxScoreGraphics receives the low level drawing commands from JMSL's score renderer. But instead of drawing to Java components, it passes these commands up to MaxScore which in turn sends messages to Max/LCD.
For example, Java's Graphics line drawing method was overridden in MaxScoreGraphics with:

```
public void drawLine(int x1, int y1, int x2, int y2) {
        if (maxScore != null) {
            maxScore.renderLine(x1, y1, x2, y2);
        }
    }
```

The code fragment above shows that the Java Graphics.drawLine() command is passed up to a reference to MaxScore object. MaxScore defines renderLine() as follows, providing the conduit to Max:

```
void renderLine(int x1, int y1, int x2, int y2) {
     String cmd = "linesegment";
     outlet(0, new Atom[] {
         Atom.newAtom(cmd),
         Atom.newAtom(x1),
         Atom.newAtom(y1),
         Atom.newAtom(x2),
         Atom.newAtom(y2)
         });
    }
```

To put it in somewhat anthropomorphic terms, JMSL itself has no idea that it is rendering a Score to Max. It renders to a Graphics object as it usually does, not knowing that this graphics object punts the commands off to Max via MaxScore. This makes for a very fast rendering engine, as only simple graphics commands are used (as opposed to an alternative design that might use a non-realtime rendering engine such as Lilypond). We were pleased that the MaxScore project could leverage the score notation already available in JMSL with only a few new classes required to bridge it to Max.
Scheduling of playback is handled by JMSL's hierarchical scheduler (Didkovsky, Burk, 2001). JMSL uses a network of parent/child relationships of "Composable" objects that pass timestamps up and down a scheduling hierarchy. Java threads are responsible for scheduling, but JMSL absorbs the vagaries of thread timing by implementing the notion of "advance time". As long as timing jitter varies by less than this advance time window, scheduling is solid. As such, the timestamps passed out of the MaxInstrument to the MaxMSP environment may be slightly in the future (as long as they are not in the past, timing is solid).

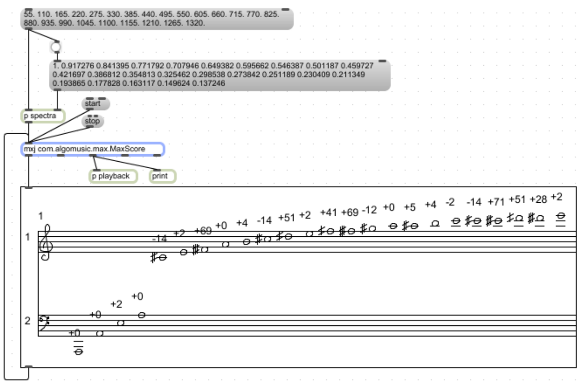
4. TWO APPLICATIONS OF MAXSCORE

Embedded in Max/MSP, an environment which has served composers and computer musicians for more than 18 years and is most likely, for its visual paradigm, the most popular music programming environment, MaxScore fills a gap in Max's functionality which up to this date offers only very rudimentary support for symbolic notation (nslider). MaxScore lends itself to a multitude of applications, of which real-time composition and notation are probably the most complex.

While music transcription of algorithmically generated music was originally done by hand (Ames), and later by notation programs such as Finale and Sibelius, or composition tools such as OpenMusic (Agon *et al*), the real challenge lies in performing such tasks in real time. Although OpenMusic (OM) and its cousin PWGL (a further development of OM's predecessor PatchWork) (Laurson, Norilo) have reached (near) real-time capability thanks to modern computers, MaxScore has the advantage of working within Max without requiring inter-application messages to be sent via MIDI or OpenSoundControl.

MaxScore behaves like a hierarchical matrix whose elements (measure, staff, track, note, interval) can be queried with specific messages such as "getMeasureInfo 0" or "getNoteInfo 2 5 1 5" (info for 6th note of the 2nd track of the 6th staff of the 3rd measure). The output is in XML format, which after parsing, can be further processed in Max. At this stage, all of the 22 fixed attributes of a note can be changed on the fly, interactively or programmatically with messages such as setPitch or setTiedOut. In addition, an unlimited number of extra *note dimensions* can be defined by the user for real-time control of sound synthesis, video processing etc. Relying on its own scheduler, MaxScore will sequence its note events and act as timeline capable of sending messages to Max much like Keith Hamel's Notability, but—once again—all within the Max environment.

Several applications can be conceived: Music can now be transcribed into rich music notation by taking secondary attributes like articulation and dynamic change into consideration. For instance, a Max patch could analyze the amplitudes of a series of notes and apply crescendo or decrescendo wedges to the notation accordingly. Aficionados of microtonal music will find the text attribute convenient, which can be utilized to show cent deviations in addition to the built-in quarter-tone accidentals. Playback of microtonal music benefits from the floating-point resolution of the pitch attribute.



```
55. 110. 165. 220. 275. 330. 385. 440. 495. 550. 605. 660. 715. 770. 825.
880. 935. 990. 1045. 1100. 1155. 1210. 1265. 1320.
```

```
1. 0.917276 0.841395 0.771792 0.707946 0.649382 0.595662 0.546387 0.501187 0.459727
0.421697 0.386812 0.354813 0.325462 0.298538 0.273842 0.251189 0.230409 0.211349
0.193865 0.177828 0.163117 0.149624 0.137246
```

Figure 4. Displaying music notation inside a Max patcher can be used for music analysis, among other applications. The bcanvas abstraction displays an overtone series OpenMusic-style with microtonal deviations printed on top of the notes.

Other types of messages are used to control music rendering: the nextPage and previousPage messages are practical when MaxScore is employed in a situation where musicians turn their pages interactively with a pedal or other means. The integration of the MaxScore canvas into a Max bpatcher, finally, permits the use of several instances within a single Max patcher window, which, among other applications, is very useful for educational purposes.

4.1. Ivresse '84

Georg Hajdu commissioned Nick Didkovsky to create MaxScore and received funding support from "Bipolar – German Hungarian Cultural Projects". Hajdu used MaxScore extensively while it was in development. The composition Ivresse '84 represents the first real-time composition realized with MaxScore and was premiered in September 2007 at the Music in the Global Village conference in Budapest, Hungary (Hajdu 2007). The composition, performed by violinist János Négyesy and the European Bridges Ensemble, is based on John Cage's first Freeman etude (Cage 1992). Before subjecting his music to an algorithmic process, the etude, which was written in space notation had to be transcribed by JMSL into standard music notation. The resulting 120 measures (with a duration of 2 seconds each) were categorized according to the similarity of their gestures.

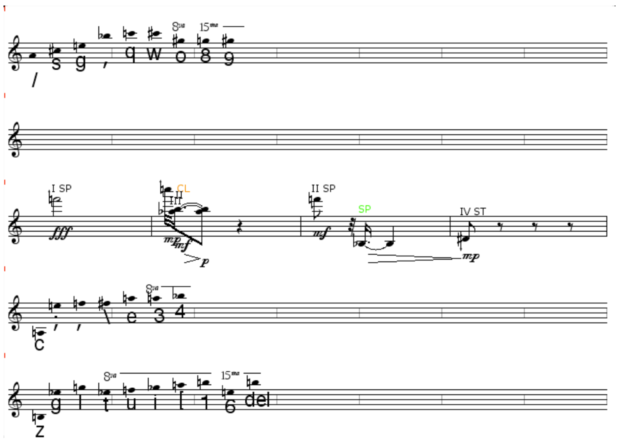


Figure 5. Excerpt from Ivresse '84 for violin and 4 laptop performers.

Based on these judgments, the composition was reassembled in real-time by a stochastic process controlled by the conductor and presented in standard music notation to the violinist who sight-read the music. Ivresse '84 employs an additional mxj object (JScoreTranslator) created by Hungarian composer Adam Siska (Siska) to bridge between MaxScore and Quintet.net, Hajdu's networked multimedia performance environment (Hajdu 2005)

4.2. Macaque

Originally designed as a bridge between MaxMSP and Lemur, the Mac OS 9 legacy software for partial-tracking analysis (Fitz, Haken), the spectral transcription software Macaque was originally created by Hajdu to transcribe partial-tracking data into standard music notation via Finale's Enigma format. The process was far from trivial and required several steps between Max and Finale before it was completed. Utilizing MaxScore's transcription method and score rendering, Macaque has become an integrated environment capable of reading SDIF 1TRC files generated by SPEAR (Klingbeil) and other applications, manipulating their content and translating the data into standard music notation.

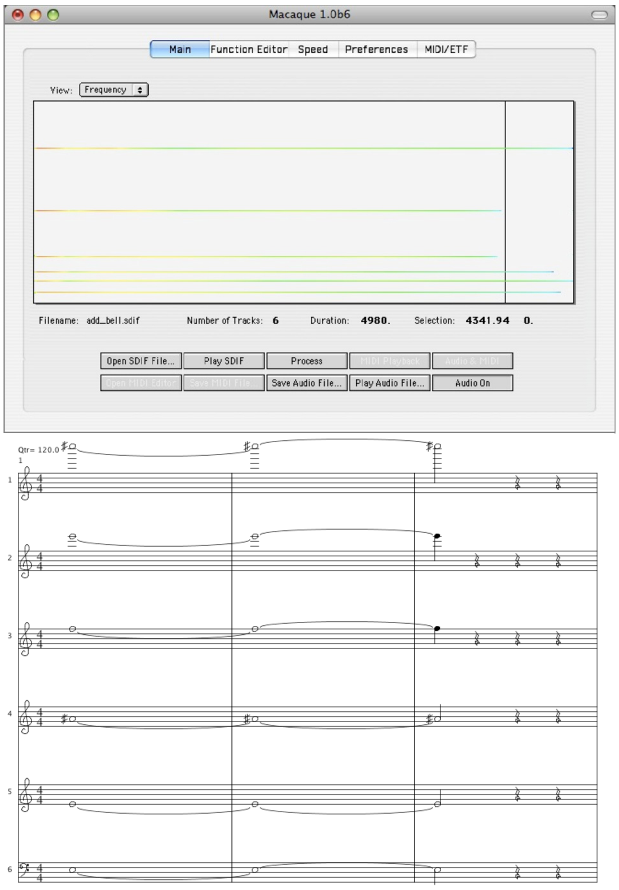


Figure 6.Macaque interface showing spectrum (top of figure) and its MaxScore transcription (bottom of figure).

5. CONCLUSION

MaxScore brings rich music notation to Max/MSP, the *lingua franca* of the computer music community. By opening the door to musicians and composers trained in reading music, MaxScore has the potential to build bridges between musical communities.

The challenge lies in building elegant systems allowing musicians to read music created programmatically in real-time off their computers, hence staging a complex social and technical interplay between performer, composer, and machine.

MaxScore is freely available for download at http://www.algomusic.com/maxscore/ .

6. REFERENCES

[1] Agon, C., Assayag, G., Laurson, M. and Rueda, C. (1999) "Computer Assisted. Composition at Ircam: PatchWork & OpenMusic", *Computer Music Journal* 23(5). 59 - 72

[2] Ames, C. (1987). Automated Composition in Retrospect. *Leonardo*, 20(2): pp. 169-185.

[3] Cage, J. 1992. The Freeman Etudes. Book 1 & 2. Edition Peters.

[4] Didkovsky, N., Burk, P.L., (2001). "Java Music Specification Language, an introduction and overview." *Proceedings of the International Computer Music Conference*, pp. 123-126.

[5] Didkovsky, N. (2004). "Java Music Specification Language, v103 update." *Proceedings of the International Computer Music Conference*, pp. 742-745.

[6] Didkovsky, N. and L. Crawford (2007). "Java Music Specification Language and Max/MSP." *Proceedings of the International Computer Music Conference*. pp. 620-623

[7] Fitz, K., and Haken, L. (1997). Sinusoidal modeling and manipulation using Lemur. *Computer Music Journal* 20(4): 44-59.

[8] Hajdu, G. (2005). Quintet.net: An environment for composing and performing music on the Internet. *Leonardo Journal* 38(1).

[9] Hajdu, G. (2007). "Playing Performers." *Proceedings of the Music in the Global Village Conference,* Budapest, Hungary, pp. 41-42.

[10] Laurson, M. and Norilo, V. (2006). "From Score-Based Approach Towards Real-Time Control in PWGLSynth," *Proceedings of the International Computer Music Conference*, pp. 29–32.

[11] Klingbeil, M. 2005. "Software for spectral analysis, editing, and synthesis." *Proceedings of the International Computer Music Conference*.

[12] Nienhuys, H., Nieuwenhuizen, J., "LilyPond, A System for Automated Music Engraving". *Proceedings of the XIV Colloquium on Musical Informatics* (XIV CIM 2003), Firenze, Italy, May 8-9-10, 2003

[13] Puckette, M., Zicarelli, D. MAX Development Package. Opcode Systems, Inc., 1991

[14] Schwarz D., M. Wright. (2000) "Extensions and Applications of the SDIF Sound Description Interchange Format." *Proceedings of the International Computer Music Conference.*

[15] Siska, A. (2007). "Pipelining between JMSL and Quintet.net." *Proceedings of the Music in the Global Village Conference,* Budapest, pp. 28-30.