

MAXSCORE – CURRENT STATE OF THE ART

Georg Hajdu

Hochschule für Musik und Theater Hamburg
Center for Microtonal Music and
Multi-Media (ZM⁴)
georghajdu@hfmt-hamburg.de

Nick Didkovsky

Rockefeller University
didkovn@mail.rockefeller.edu,
www.algomusic.com

ABSTRACT

We present recent developments of MaxScore – an mxj notation object for Max—and the environment it is embedded in. Since the first presentation of MaxScore at the 2008 ICMC in Belfast, the software has gone through some major development stages making it a prime choice for music notation within the Max and Live (via Max for Live) software environments. Besides providing a growing set of well over 230 messages, which communicate with the JMSL API, development has focused on offering convenient tools for real-time composition and notation in networked music environments as well as graphical and microtonal notation. The LiveScore/MaxScore editors are central to our efforts—implementing different microtonal notation modes (48TET, 72TET and Just Intonation) as well as the Max patcher-based Scorepions plugin system.

1. INTRODUCTION

MaxScore is a Java Max object developed by Nick Didkovsky, which works in conjunction with Max abstractions created by Georg Hajdu [1][2]. Being released in 2007 (with a precursor from 2005), it represents the first integrated notation solution for the Max programming environment and has enjoyed a loyal following despite powerful alternatives such as the Bach Project (<http://www.bachproject.net>). MaxScore renders to the Canvas abstraction, which consists of a set of nested Max patches (Fig. 1). While earlier versions of MaxScore have used basic drawing shapes such as line segments, polygons and arcs to represent musical signs, the versions since 2009 employ a music font and use a more abstract definition of music glyphs (e.g. note heads and clefs) as well as music curves (e.g. ties, slurs, tuplets, ottava alta/bassa). Since rendering is done in Max, music glyph and curve messages can be intercepted and reinterpreted by the Canvas, allowing for greater flexibility and adaptability compared to “boxed” solutions. The Canvas abstraction now accommodates various music font maps (New Aloisen, Jazz, Maestro and Opus) and microtonal maps (for eighth-tone, twelfth-tone and Extended Helmholtz-Ellis JI Pitch notation) and is also capable of exporting vector graphics in svg format.

Drawing is done with the Max lcd object to which the processed music glyph and curve messages are ultimately passed. These come out of MaxScore’s first outlet while its second outlet is also used in response to info messages which play an important role in some scenarios in which note and/or staff attributes need to be queried before a music glyph is drawn. The third and fourth outlets pass instrument output and sequence dumps as well as echo the indexes of the notes transcribed by the transcriber (Fig. 2).

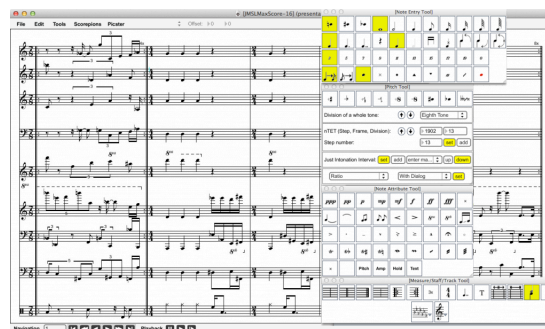


Figure 1. LiveScore Editor with floating palettes.

MaxScore employs a hierarchical data format where score, measure, staff, track, note and interval elements (among others) form a tree. New attributes (e.g. for the visibility of notes, noteheads or stems etc.) are continually being added to accommodate new scenarios. In addition, a user can define an unlimited number of extra note dimensions which can be used to control various processes and drive playback devices.

To complement the functionality of MaxScore, Ádám Siska has designed a number of Max externals: *sadam.rapidXML*, *sadam.base64* and *sadam.lzo* among others [3]. As some crucial info queries return XML code, Siska’s Max wrapper for the efficient RapidXml parser is an indispensable ingredient in most scenarios involving real-time notation, while the other two objects facilitate the communication of remote MaxScore instances—a welcome feature in networked music performances.

Recently, a searchable dictionary was added in order to facilitate programming with MaxScore with its plethora of different messages.

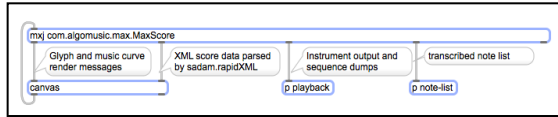


Figure 2. Communication between the main components of a MaxScore patch.

2. LIVE

When Max for Live extension was released in 2010 by Ableton and Cycling '74, it became obvious that MaxScore could provide what numerous Live users were desiring: a device for the notation of Live MIDI clips. To this end, we have developed two devices for display (LiveScore Viewer) and editing (LiveScore Editor), and added two more devices (a sampler and a soundfont player) for microtonal playback, which is not natively supported by Ableton.

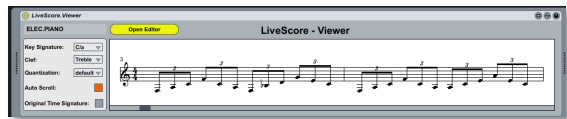


Figure 3. Screen shot of the LiveScore Viewer

2.1. LiveScore Editor

While the LiveScore Viewer was designed to give users a quick and musically sensible overview of a clip's content, the editor makes use of the Live API for bi-directional communication: a clip can be selected from the Live clip matrix, transcribed and edited with all edits being instantaneously mirrored in the original. Before transcription, a key finder (utilizing a neural network recognizing major/minor key profiles [4]) and a clef finder (analyzing the pitch distribution in the clip) preprocess the note lists. Additionally, a percussion map translates MIDI key numbers into the appropriate pitches and symbols should percussion notation be desired (Fig. 6).

A set of menus and floating palettes are available for convenient editing and embellishing of the transcribed clips, which can be saved in JMSL's XML score format (Fig. 1). Staff attributes can be set in the Staff Manager, which represents an alternative view of the Editor window.

Since for a given clip, LiveScore's transcriber will only transcribe one track (i.e. voice) and shorten the duration of a note when a new note starts, the question arises as to how it preserves the polyphony of overlapping note events contained in a Live clip. The answer is as follows: note elements have a "hold" attribute which stores sustain time and will be used when the Live clip is being updated.

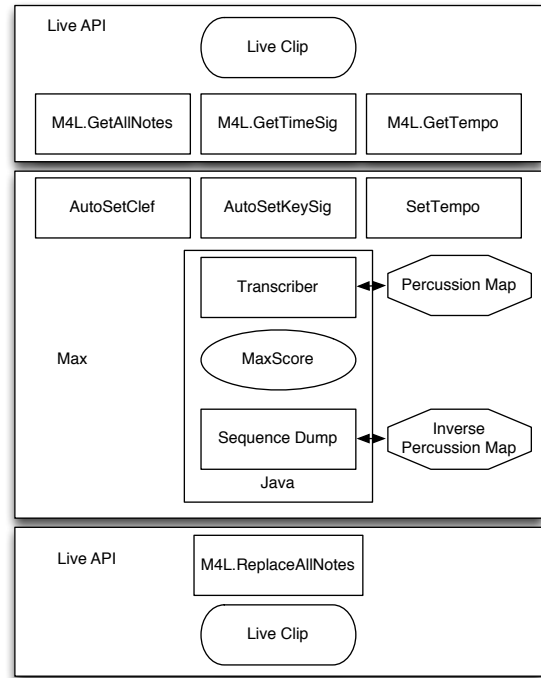


Figure 4. Communication between Live, Max and MaxScore during transcription and updating. The flow is top to bottom.

Ableton Live (pitch time duration velocity muted_flag)					
notes 13					
note	59	42.166668	0.158333	125	0
note	60	42.	0.158333	125	0
note	60	42.333332	0.158333	125	0
note	62	42.5	0.158333	125	0
note	64	42.666668	0.158333	125	0
note	65	42.833332	0.158333	125	0
note	67	43.	0.158333	125	0
note	69	43.166668	0.158333	125	0
note	70	43.333332	0.158333	125	0
note	72	43.5	0.158333	125	0
note	74	43.666668	0.158333	125	0
note	76	43.833332	0.158333	125	0
note	77	44.	3.	126	0
done					

MaxScore (instrument time tempo duration pitch velocity hold event_flag additional_note_dimension)								
sequenceDump start								
0.	0.	60.	1.	0.	0.	1.	3.	0.
...
0.	42.	60.	0.166667	60.	125.	0.158333	3.	0.
0.	42.166668	60.	0.166667	59.	125.	0.158333	3.	0.
0.	42.333332	60.	0.166667	60.	125.	0.158333	3.	0.
0.	42.5	60.	0.166667	62.	125.	0.158333	3.	0.
0.	42.666668	60.	0.166667	64.	125.	0.158333	3.	0.
0.	42.833332	60.	0.166667	65.	125.	0.158333	3.	0.
0.	43.	60.	0.166667	67.	125.	0.158333	3.	0.
0.	43.166668	60.	0.166667	69.	125.	0.158333	3.	0.
0.	43.333332	60.	0.166667	70.	125.	0.158333	3.	0.
0.	43.5	60.	0.166667	72.	125.	0.158333	3.	0.
0.	43.666668	60.	0.166667	74.	125.	0.158333	3.	0.
0.	43.833332	60.	0.166667	76.	125.	0.158333	3.	0.
0.	44.	60.	3.	77.	126.	3.	3.	0.
0.	47.	60.	1.	0.	0.	1.	3.	0.
sequenceDump stop								

Table 1. Comparison between Live note lists and MaxScore sequence dumps

Still, bi-directional communication between Live and MaxScore is challenged by an important issue which still awaits a solution: Live's note list only consists of 5 parameters (pitch, time, duration, velocity and a *muted* flag), not enough to either store or reference additional attributes such as dynamics, articulations, slurs etc. Pitch is stored as an integer value, which won't allow for microtonal deviation either [5]. Therefore, the creation of rich scores in LiveScore quickly becomes a one-way street where an embellished clip lives on outside the Live set, separated from its twin clip. We have brought this issue to the attention of Ableton, and are optimistic that a solution will be found in the future.

There are two playback modes: (1) Live's playback which sends the clips' note events to standard Live devices via the internal MIDI bus and is synced to MaxScore's page turns and note flashes as well as (2) MaxScore's playback engine which passes score events to a capable audio device via a non-standard format which was devised to accommodate microtonal pitches. In order to synchronize with the Max and Live environments, JMSL's scheduler is now being driven by an implemented JMSLMaxClock, used as JMSL.clock [6].

While developing the software, special attention was given to allowing several instances of the editor to coexist in one Live set. This particularly concerned potential clutter caused by using several instances of the same floating palettes (Fig 1.). We have implemented a mechanism that prevents a second instance of a palette to be opened and dynamically directs user actions to the most recent, front-most editor window, regardless of whether the palettes are children of this window or not.

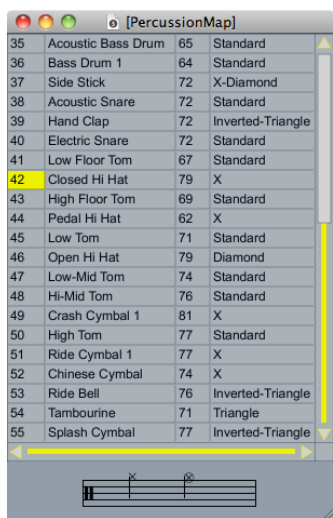


Figure 5. The Percussion Map uses another MaxScore instance to display pitch and notehead for a given percussion instrument.

2.2. Picster

Since MaxScore's repertoire of score markings is rather basic, we added a new feature called "rendered messages" making the repertoire of text markings and graphics virtually unlimited. These messages, which can be either attached to notes, measure/staves or measures,¹ consist of an index, position information and a string, which will be passed to the drawing engine (the Canvas abstraction in our case). As long as the string contains messages understood by the engine, they will be rendered and displayed in the score. Rendered messages will be inserted and saved in JMSL scores marked up by various UserBean tags.

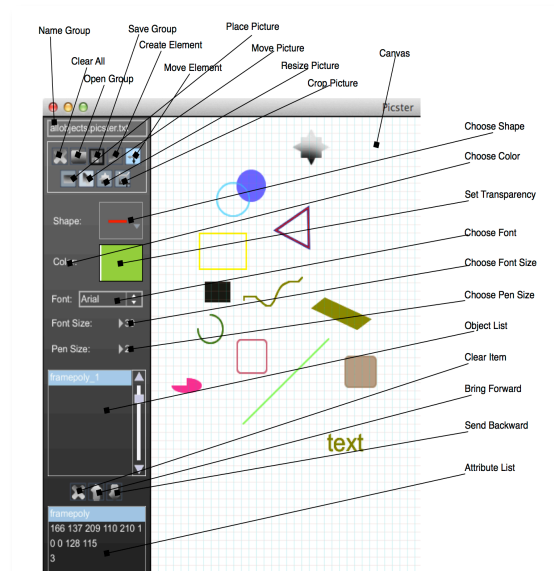


Figure 6. Screenshot of the Picster GUI with a sample of different shapes

Picster is a Max patch created by Jacob Sello and one of us (Georg Hajdu) providing a drawing environment, where rendered messages can be interactively created and edited. Users can choose from 14 different shapes (from linesegments, ovals, rectangles and polygons to freehand drawings, text and scalable pictures). Markings and graphics can easily be added to a library, accessible from the Editor's Picster menu. Once created, the objects the rendered elements are attached to, can be identified and highlighted by shift-ctrl-click, and the elements themselves be moved or re-edited in the Picster patch.

¹ Measure and staves form a two-dimensional matrix with staves on the x axis and measures on the y axis. A measure refers to vertical content across all staves for a given measure number, whereas staff refers to horizontal content across all measures for a given staff number. A measure/staff refers to the cross-section between a given measure and staff. This is the smallest unit in the matrix.

2.3. Scorepions

Scorepions² are Max patches consisting of MaxScore messages, which reside in a special folder inside MaxScore Lib and may be dynamically invoked by their parent patch—thus forming a plugin system similar to those available in most commercial music editors. Scorepions can be used to programmatically generate and process all elements of a score. Amongst the 15 plugins that are currently part of the MaxScore release, the *DJster-Autobus* Scorepion deserves special attention as it represents a revived and further developed version of Clarence Barlow’s legacy software AUTOBUSK—a program devised for algorithmic composition (<http://www.musikwissenschaft.uni-mainz.de/Autobusk/>). The *DJster-Autobus* Scorepion takes the non-real-time part of Barlow’s program and uses its algorithms to interactively fill measures with note events based on given parameter presets. His application has also been extended to accommodate the microtonal scales from the Scala archive (an archive with well over 4000 scales; <http://www.huygens-fokker.org/docs/scales.zip>) as well as complex additive meters. A detailed explanation of the underlying algorithms and their adaptation to Max is outside the scope of this paper.

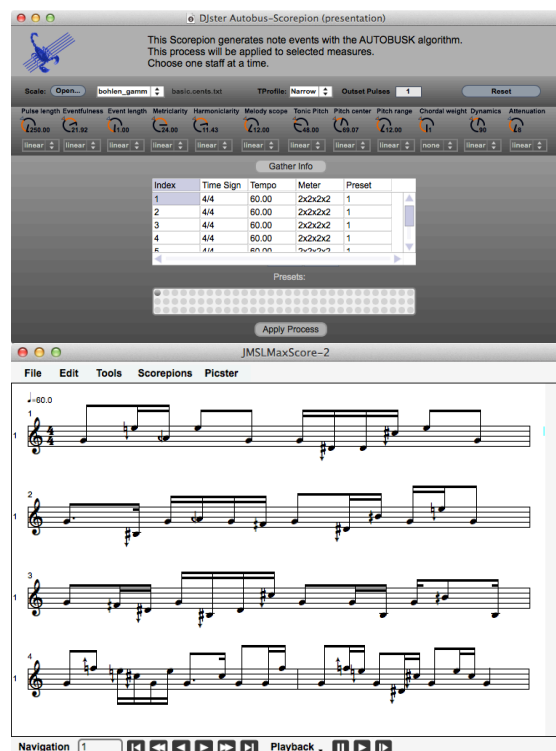


Figure 7. GUI of the “Djster Autobus” Scorepion (top). Resulting five measures algorithmically composed in the Bohlen-Pierce Gamma scale (bottom).

2.4. Microtonality

JMSL natively supports quarter-tone notation. All other types of notation (8th-tone, 12th-tone, Just Intonation) require remapping of music glyph messages in the Canvas. As an example we will now describe the implementation of the Extended Helmholtz-Ellis JI pitch notation, which by far surpasses other microtonal maps in terms of complexity and computational resources.

The Extended Helmholtz-Ellis JI Pitch notation was developed by Marc Sabat and Wolfgang von Schweinitz as a comprehensive approach to the notation of music in Just Intonation (JI) [7]. The aim is to encode harmonic relationships between notes in terms of their accidentals. Sabat and von Schweinitz have created the HE font with accidentals capable of representing a 61-limit harmonic space, but have themselves suggested to cap the system at 23-limit (or possibly even lower). Since JMSL has no built-in concept of harmonic relationships, ratios expressing these relationships need to be calculated each time a page is being rendered. Two attributes are taken into consideration: the key signature of a measure representing the fundamental as well as the pitch of a note in floating point precision. A lookup table is being used to determine the ratio corresponding to the cent interval between the fundamental and the pitch. Lookup tables can be generated according to different principles such as sensory consonance, harmonic entropy or, in our case, harmonic energy, a measure derived from Clarence Barlow’s concept of the harmonicity of intervals—which proves to do justice to most musical situations [8].

Once a ratio is found for the given cent interval, its numerator and denominator are subject to prime factorization and an algorithm is being applied to determine the enharmonic spelling of the note and its accidentals.

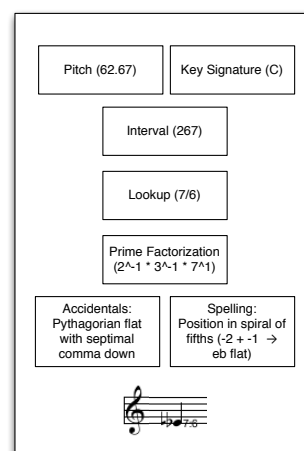


Figure 8. Accidental-finding algorithm for the Extended Helmholtz-Ellis JI Pitch Notation

² Kudos to my student Daniel Dominguez for suggesting its name.

To specify the position and spelling of the note, the fundamental is being used as the point of departure on a non-closing spiral of fifths, with the numerator

and denominator determining the direction (clockwise, counterclockwise) and amount (in terms of fifths) to move. Accordingly, up to three accidentals are being chosen in accordance to the primes contained in the ratio.

Ultimately, the growing repertoire of LiveScore's microtonal modes will allow users to create different representations of the same music either by instantaneously switching between views or by juxtaposing different ones. In an unpublished talk at the 2010 Bohlen-Pierce symposium in Boston, Hajdu showed a Max patch, capable of several representations of the same music written in the Bohlen-Pierce tuning. He coined those representations *cognitive notation* (notation in familiar contexts such as five-line diatonicism), *instrumental notation* (notation based on the physicality of an instrument, e.g. tablatures) and *logical notation* (notation capable of visually representing equal distances) (Fig. 9).

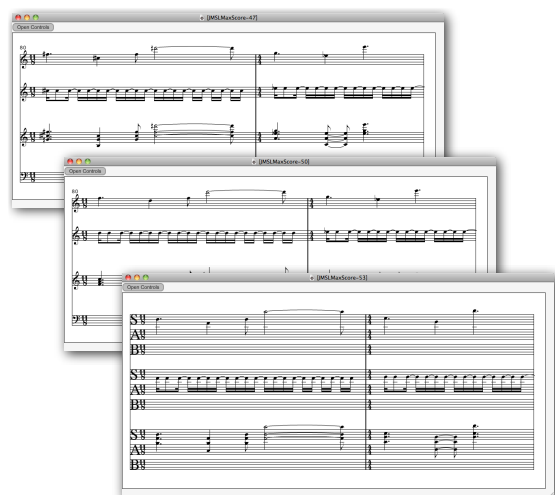


Figure 9. Alternate views of the same music (Beyond the Horizon for 2 BP clarinets and synth by Georg Hajdu) in Bohlen-Pierce tuning (top: cognitive notation; middle: performance notation; bottom: logical notation)

2.5. Printing

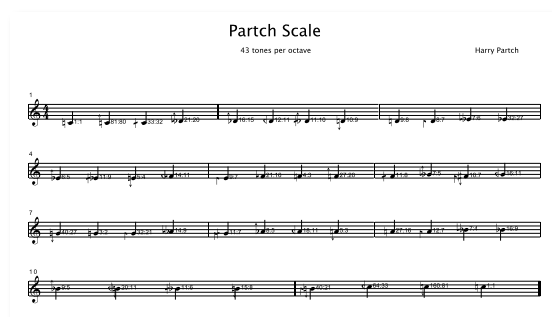


Figure 10. Rastered svg file of the 43-tone Partch scale in Extended Helmholtz-Ellis JI Pitch Notation with ratio labels.

It is possible to create pdfs directly from the Editor. Two Java applications have been integrated via shell scripts (Mac OS X) or DOS commands (Windows): the Batik Rasterizer which turns svg files into bitmapped pdf files and PDF.jar which combines separate pdf pages into a single document.

2.6. MaxScore Editor

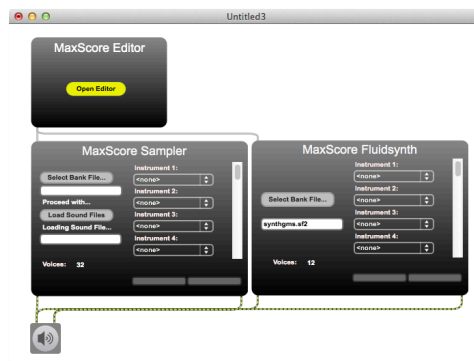


Figure 11. A Max patcher can be used to connect MaxScore components (editor and two playback devices)

At some stage in the development of LiveScore it became obvious that it would make sense to offer the editor's convenience in the Max/MSP patching environment. The MaxScore Editor (a Max bpatcher) has the same features as its sibling but can also be connected with other instances of the Editor, locally or over IP networks, as well as with synthesizers and other playback devices. It also reacts to the same set of messages as the core MaxScore object. Hajdu's composition Swan Song is an example for using MaxScore Editor in a live performance with musicians reading their music off the screens of networked computers.

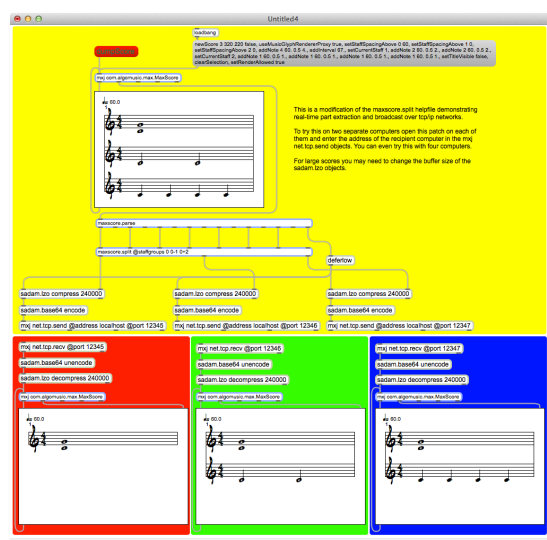


Figure 12. Example patch for using MaxScore to send scores over an IP network.

3. EXAMPLES

The MaxScore homepage features a section dedicated to music written by various composers, Nicolas Collins, Arne Eigenfeldt and Peter Votava, among them [9]. In the following paragraphs, we will present the piece *Swan Song* by Georg Hajdu, which was premiered at the 2011 Shanghai Electronic Music Week. An earlier piece of his, *Schwer...unheimlich schwer* for bass clarinet, viola, piano and percussion has been analyzed in an issue of the Contemporary Music Review on Network Music [10].

3.1. Swan Song

Like some of Hajdu's earlier pieces, *Swan song* - 送别 for cello and percussion is based on transcriptions of preexisting sonic materials: speech, music and noises. For this piece Hajdu chose the final scene of a masterpiece of Chinese cinema called *Farewell, My Concubine* by Chen Kaige, a movie that had a great impact on him when it was released in 1993. The movie revolves around a complicated love story and features scenes from an eponymous Peking Opera. Life and theater blend dramatically in the final scene.



Figure 13. Section from *Swan Song* with non-standard glyphs inserted via "rendered messages".

The rendering of the transcribed materials by the cello and percussion, mimicking the voices and instruments of Peking opera, are accompanied by processed video from the movie as well as electronic and prerecorded sounds. The first two tracks of the master score are being used for real-time part extraction and sent to the players over the network, the third and fourth tracks for the control of audio and video playback and the fifth is a click track, synchronizing the musicians to the audio and video playback (Fig. 13). Like in *Schwer...unheimlich schwer* the musicians read their music off of computer screens, only in this

case, the entire score is being sent at the beginning of the piece with the players using an AirTurn Bluetooth Page Turner to turn their pages.

4. ONLINE PRESENCE AND LICENSING OPTIONS

We are maintaining a WordPress CMS at <http://www.computermusicnotation.com> to promote MaxScore on the WWW. The website features pages for news, downloads, documentation, support, projects and is connected to a mailing list as well as a discussion forum.

We are offering two JMSL license options at different prices: A JMSL license as well as the lower priced LIVE license, which will disable all Java-only features such as the standalone score editor and the JSyn API.

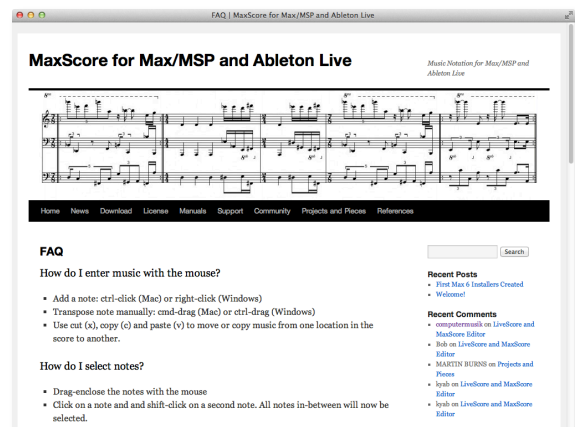


Figure 14. Screenshot of computermusic.com

5. OUTLOOK

Since its first release in 2007, MaxScore has developed into an environment for music notation with considerable versatility and adaptability. The recent release of Max 6 opens an opportunity for a review of the existing code base. Two developments are particularly worth mentioning: JavaScript and dict. Cycling '74 has finally created a technology called mgraphics offering an alternative to the 20 year old lcd object by including a multitude of new drawing modes (including Bézier curves) and svg support at a comparable speed. Tapping into the power of the Mozilla JavaScript engine we will consolidate the nested *Canvas* and *Picster* abstractions into large jsui objects.

The dict package consists of a set of objects supporting a hierarchical data structure. By switching MaxScore's XML output to dict's JSON format we are expecting speed and efficiency gains when exchanging and processing score data. We

are also working on MusicXML import, which will add to MaxScore's user-friendliness.

6. REFERENCES

1. Nick Didkovsky & Georg Hajdu (2008). MaxScore. "Music notation in Max/MSP". *Proceedings of the International Computer Music Conference*.
2. Georg Hajdu & Nick Didkovsky. "On the evolution of music notation in network music environments". *Contemporary Music Review* 28, 4/5, pp. 395 – 407
3. <http://www.sadam.hu/?q=software> (retrieved on February 14, 2012)
4. Carol L. Krumhansl. *Cognitive Foundations of Musical Pitch*. New York: Oxford University Press, 1990.
5. http://cycling74.com/docs/max6/dynamic/c74_docs.html#m4l_live_object_model (retrieved on February 14, 2012)
6. <http://www.didkovsky.com/JavaMusicSystems/JMSL3.pdf> (retrieved on February 14, 2012)
7. Marc Sabat, "The Extended Helmholtz-Ellis JI Pitch Notation," in *Mikrotöne und Mehr*. von Bockel Verlag, 2005, pp. 315–331.
8. Clarence Barlow (1987). "Two essays on theory". *Computer Music Journal*, 11, 44–60.
9. http://www.computermusicnotation.com/?page_id=266 (retrieved on February 14, 2012)
10. Georg Hajdu, Kai Niggemann, Ádám Siska & Andrea Szigetvári. "Notation in the Context of Quintet.net Projects". *Contemporary Music Review*, 29, 1, pp. 39 – 53