The “Cantus Index GUI”: A Visual Interface for the Creation of Electronic Plainchant Resources

Morné BEZUIDENHOUT – Mark BRAND

University of Cap Town – University of Port Elizabeth
South Africa

Abstract: The “Cantus Index GUI” (Bezuidenhout) is a data-capture utility that facilitates the population of a database containing electronic indices for the Western plainchant manuscripts in the Grey Collection of the National Library of South Africa. It seeks to accelerate to capture of data describing plainchant sources, and to encourage standardization with a view to future consolidation and analyses of this data. The addition of the MIDINeume ActiveX Control (Brand) points the way to some of the expansion possibilities.

Keywords: plainchant resources, electronic index

Introduction

This paper discusses the “Cantus Index GUI”, a research utility authored by Morné Bezuidenhout. It further includes a separate discussion of the “MIDI-Neume ActiveX Control” by Mark Brand, a custom component created for, and integrated into the “Cantus Index GUI”.

The “Cantus Index GUI”

The “Cantus Index GUI” is a data-capture utility that facilitates the population of a database containing electronic indices for the Western plainchant manuscripts in the Grey Collection of the National Library of South Africa.[1] It allows for the input of chant data from the sources of the liturgical Office that are in the Grey Collection. The design of a version for the Mass manuscripts will only begin on completion of the processing of the Office manuscript data. The primary aim of this project is to produce utilities that will bring some

[1] For descriptions of the manuscripts and literature about the Collection see: F. C. Steyn (ed.), *The Medieval and Renaissance Manuscripts in the Grey Collection of the National Library of South Africa, Cape Town*, Analecta Carthusiana 180, Salzburg: Institut für Anglistik und Amerikanistik, 2002.

relief to the tedium of capturing not only the Grey data but also data in other collections of plainchant manuscripts.

The design of a particular Grey database utility always proceeds from an investigation into the availability of existing databases. As a point of departure, the design of a utility attempts to incorporate the design of existing models with preference being given to those that have well-maintained data stores. In the case of Office manuscripts, the CANTUS database, maintained at the University of Western Ontario, is an obvious choice.[2] This database is a long-standing project with a well-evolved and familiar data-file structure that also offers a large and growing data pool of chants in sources of the liturgical Office. Both the data-file structure and data of the CANTUS database play a significant role in the operation of the "Cantus Index GUI".

This paper does not provide a complete description of the functionality of the GUI. It concentrates mainly on the interaction of the GUI with existing electronic resources for the study of plainchant. The reader should consult the web site for more detailed information about the GUI.[3] In the following, we consider how the GUI adapts the file structure prescribed for CANTUS data files for its own storage purposes. We then continue with a discussion of the role that the existing CANTUS data pool plays in the creation of the GUI data store and conclude with a look at the GUI's implementation of automation and an electronic version of *Corpus Antiphonalium Officii*.[4]

Data storage in CANTUS and the "Cantus Index GUI"

The data in the CANTUS database is contained in ASCII files with fixed line lengths.[5] They are known as *CANTUS indices*. Each ASCII file in the database contains chant data for a single Office manuscript, typically an Antiphoner, and each line in this file, that is a line of text ending with a carriage return, represents data from one particular chant. While each line of text is unique as a whole, there is a large degree of redundancy (duplication of data) within and

[2] http://publish.uvo.ca/~cantus/. The entire web site is included on: Center for the History of Music Theory and Literature (CHMTL), *Thesaurus Musicarum Latinarum; saggi musicale italiani; CANTUS: A Database of Gregorian Chant*, Bloomington: Indiana University Press (CDROM). The data on the 2002 edition of the CDROM includes 70 Office manuscripts and reflects the state of the CANTUS data up to about June/July 2002. CANTUS announced the addition of data from the 71st manuscript in August 2002. Unless otherwise stated, information about CANTUS in this paper is taken from the web pages published on the CDROM.

[3] http://www.musictechnology.co.za/plainchant/.

[4] R.-J. Hesbert (ed.), *Corpus Antiphonalium Officii, vol. III, Invitatoria et Antiphonae*, Rome: Herder, 1963; R.-J. Hesbert (ed.), *Corpus Antiphonalium Officii, vol. IV, Responsoria, Versus, Hymni et Varia*, Rome: Herder, 1970.

[5] Contributors can submit CANTUS indices with 85, 88, or 131 characters (columns) per line. The GUI uses only the 131-column files.

among the CANTUS indices.[6] The reason for the inherent redundancy lies in the models that inspired the CANTUS project. The content of the CANTUS indices is similar to that of the paper-based indices that often accompany facsimile editions of plainchant manuscripts. The main function of the CANTUS indices is then also the same as that of its paper-based counterparts: they facilitate access to the original plainchant sources. The purpose of such an index is not to replace the need for the original source, but rather to bring the researcher closer to the original material that the data in the index represents. While one may learn a lot about a particular manuscript from such an index, especially if it is in a highly manipulable electronic format, the index does not fully represent the actual objects of study: the complete chants with text and music in their original written state.

As its name implies, the "Cantus Index GUI" assists with the creation of electronic indices for sources of the Office liturgy and specifically indices that are compatible with the CANTUS data-file structure. Lacoste recommends that indexers should create CANTUS indices with a text editor or word processing program. She remarks quite correctly that database "programmes are useful for manipulating the completed index but may be difficult to use in the creation of the index."[7] The "Cantus Index GUI" is an attempt to address this problem by providing a utility that allows the user to create indices for chant manuscripts in a database environment that is less daunting for some than, for example, Microsoft's Access database program.

The GUI creates Access databases and stores its data in tables that reside in these databases while providing the data-capture operator with a user-friendly environment designed specifically for the input and output of data from Office manuscripts. GUI users that are familiar with the CANTUS database should therefore be aware of the fact that at the most basic level of data storage, the data does not exist in independent files, but rather in tables which are part of a single file known as an Access database. The help and information documentation accompanying the "Cantus Index GUI" clarify these matters.

As ASCII files with fixed line lengths, CANTUS indices are in a format that can readily be imported into an Access database, as long as the CANTUS

[6] CANTUS is therefore not a relational database. There are of course many arguments that one could, from the point of view of database design, level against this format. If, however, one considers that, in the case of CANTUS, we are dealing with a database in the making, reliant mainly for new data upon the work of researchers, who have access to different levels of technology and who are geographically removed from one another, the current format has distinct advantages over a relational design. A relational database would require, for data-capturing purposes, access to pre-existing master database tables. In the case of the CANTUS database, the data files exist independently of each other and one can create new data files with limited access to technological resources, and reliance on the availability of existing data files.

[7] D. Lacoste, *The Cantus Database: Procedures for Indexers*, London, Ontario: The University of Western Ontario, 2002, 1.

file description is available for the definition of the table schema. Table 1 provides the schema of the GUI's table structure. The design proceeds from a conversion of the CANTUS 131-column file description into Access table properties. CANTUS users will be familiar with much of the information provided here. The CANTUS file description contributes to sixteen of the eighteen fields in *Table 1*. The additional fields, *id* and *midi*, provide for a unique identification number for each record in a particular table and the addition of melodic data.

Table 1: "Cantus Index GUI" table schema

CANTUS	**GUI field**	**Data type**	**Length**	**Allow Null**	**Key**
	id	auto-increment nr.		no	primary
Marginalia, Other information	*marginalia*	character	3	yes	**
Folio and Side	*folio*	character	4	yes	**
Sequence Number	*sequence*	character	2	yes	**
Liturgical Occasion	*feast*	character	20	yes	
Office	*office*	character	2	yes	
Genre	*genre*	character	1	yes	
Position	*pos*	character	3	yes	
Incipit	*incipit*	character	29	yes	
CAO Concordances	*cao*	character	12	yes	
Chant Identification Number	*chantid*	character	8	yes	
Mode	*mode*	character	2	yes	
Differentia	*differentia*	character	2	yes	
Extra	*extra*	character	5	yes	
Liturgical Occasion Code	*feastid*	character	8	yes	
Siglum	*siglum*	character	20	yes	
Addendum	*addendum*	character	10	yes	
MIDI	*midi*	binary		yes	

** *folio + sequence + marginalia:* alternate compound key

The *id* field makes provision for a flexible approach to the data-capture process. The GUI allows a data-capture operator to provisionally store empty fields that could possibly contain data on the completion of the index. Since the null value is the obvious value for fields that contain an unknown value in a database table, all the fields taken from CANTUS in the GUI tables allow for the inclusion of null values during the data-capturing process. As a result, it does however become possible to create duplicate records during data capturing, even though each record in a completed index will eventually be unique.

Without going into the technical details, the presence of such duplicate records in a database table is bad practice. The way in which to prevent duplicate records is to have a field that is unique for each record in the table. In the absence of such a field, it should at least be possible to construct a unique composite field group. In the absence of the latter option, it becomes necessary to add a field with unique data to the table. In the GUI, the automatically incremented unique number in the *id* field ensures that each record in a particular table is unique regardless of the data in the other fields. It is important to consider the following features of this field: First, it operates in the background and is of no concern to the data-capture operator or end user of the Grey database; and second, although it is appended to data imported from outside, its output depends on the type of data output that the end user requires. In other words, output from the "Cantus Index GUI" intended for the CANTUS database will exclude this field.

The addition of melodic data has been another major consideration. The discussion of this feature of the GUI appears below in the section devoted to the MIDINeume control. As with the *id* field, the inclusion or exclusion of the data output from this field will depend on the requirements of the end user. One should also note that this is not a required field for the GUI's databases, leaving its inclusion or exclusion in the hands of the data-capture operator.

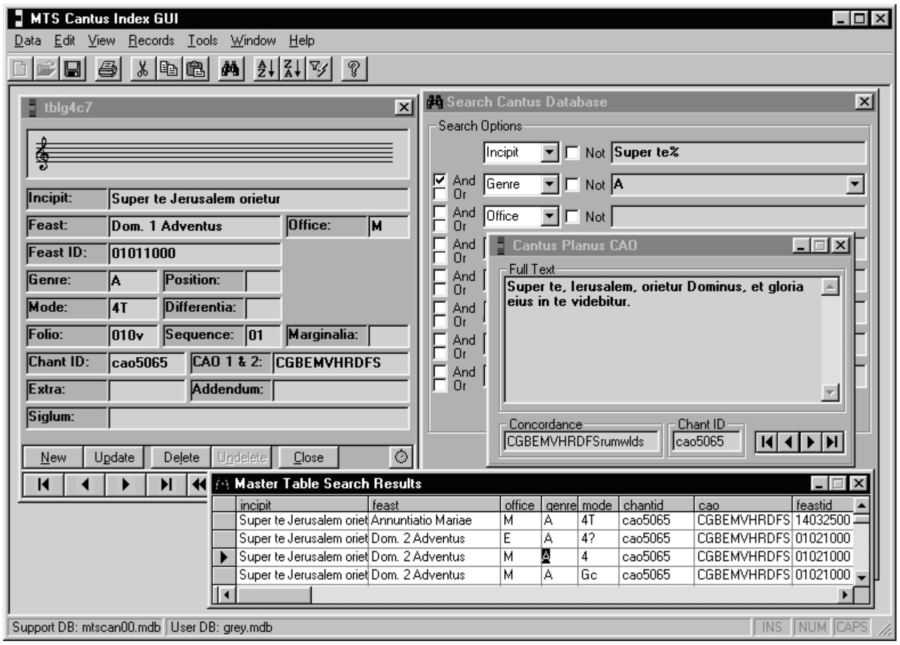
The use of existing data

At present, the CANTUS database contains indices for the plainchant contents of 71 Office manuscripts. If one considers the large amount of material that is common among Western plainchant manuscripts, it is clear that every data-capturing exercise will result in a major exercise in data-capture duplication. It follows that, as a surrogate of the CANTUS database, the "Cantus Index GUI" will perpetuate and increase the degree of redundancy among electronic plainchant indices.

The solution to the apparent problem of redundancy is a matter for careful future consideration that the "Cantus Index GUI" does not yet address. What the GUI does address is the process whereby the data-capture operator creates the redundancy. The main function of the GUI is the elimination of the physical retyping of redundant data in a new index. It does this by providing a search engine that allows the data-capture operator to easily locate redundant data within an existing pool of data. This pool can consist of any pre-existing data taken from the CANTUS database. Once the GUI locates redundant data, its inclusion into a new table is a simple click of the mouse or single keystroke on the computer keyboard.

Figure 1 illustrates the use of existing data during the data-capture process. In this example, the user has searched through CANTUS for a record containing the beginning of the antiphon text "Super te Jerusalem" (the "Search Cantus Database" window). The search then displayed the results in the "Master Table Search Results" window. In order to ensure that the *CAO* identification number ("cao5065") was correctly identified for the incipit, the user then launched a search through the electronic *CAO* (see below) that displayed the full text of the chant in the "Cantus Planus CAO" window. After comparing the *CAO* full text with the original manuscript and confirming that this was indeed the full text that appeared in the original, the user then selected, with a click of the mouse, relevant cells from the search results window for inclusion in the new table. For the role that automation would play during such a data-capture process, see section "Automation and the electronic CAO".

The target of the search through the CANTUS data is a table residing in a separate database provided with the GUI. This database, known as the *support database*, contains all of the additional tables that support the data-capturing process. One can also create this table with a utility that resides in the GUI. With this utility the user may select any number of CANTUS indices. Thereafter, the utility runs a union query that effectively removes complete redundant records and excludes fields (for example *folio, sequence* and *marginalia*) that are unique to a specific manuscript. The operation of the utility is simple with


MTS Cantus Index GUI
Data Edit View Records Tools Window Help
tblg4c7
New Update Delete Undelete Close
Search Cantus Database
Search Options
Cantus Planus CAO
Master Table Search Results

Incipit:	Super te Jerusalem orietur
Feast:	Dom. 1 Adventus
Office:	M
Feast ID:	01011000
Genre:	A
Position:	
Mode:	4T
Differentia:	
Folio:	010v
Sequence:	01
Marginalia:	
Chant ID:	cao5065
CAO 1 & 2:	CGBEMVHRDFS
Extra:	
Addendum:	
Siglum:	

Search Options: Incipit ☐ Not Super te%; ☑ And ☐ Or Genre ☐ Not A; ☐ And ☐ Or Office ☐ Not

Full Text: Super te, Ierusalem, orietur Dominus, et gloria eius in te videbitur.

Concordance: CGBEMVHRDFSrumwlds Chant ID: cao5065

	incipit	feast	office	genre	mode	chantid	cao	feastid
	Super te Jerusalem oriet	Annuntiatio Mariae	M	A	4T	cao5065	CGBEMVHRDFS	14032500
	Super te Jerusalem oriet	Dom. 2 Adventus	E	A	4?	cao5065	CGBEMVHRDFS	01021000
▶	Super te Jerusalem oriet	Dom. 2 Adventus	M	A	4	cao5065	CGBEMVHRDFS	01021000
	Super te Jerusalem oriet	Dom. 2 Adventus	M	A	Gc	cao5065	CGBEMVHRDFS	01021000

Support DB: mtscan00.mdb User DB: grey.mdb INS NUM CAPS

Figure 1: "Cantus Index GUI" with CANTUS and *CAO* searches

the user's only task being the selection of the indices. In this way users can update the data pool should new indices become available, or make a customized data pool.

Automation and the electronic *CAO*

The GUI provides for automation in the case of two sets of related fields: 1) *feast* and *feastid*; and 2) *incipit, cao* and *chantid*. The support database contains a table of the legitimate CANTUS feast names and feast codes. These link to the current index in such a manner that the entry of either one of these fields will automatically generate the other. In addition, the GUI provides the user with search-aware dropdown lists of the feast names and feast codes.

Data entries for the *incipit, cao* and *chantid* fields follow a similar procedure. This functionality relies on the presence of the electronic *CAO*, which also resides in the support database. The electronic *CAO* provided with the GUI includes the following fields: 1) the *CAO* number of a particular chant following the CANTUS conventions; 2) the incipit of this chant, again following CANTUS conventions; 3) the full text of the chant, as it appears in the electronic version of *CAO* that resides on the Cantus Planus web site at Regensburg University;[8] 4) the concordance entry for the chant given in the Regensburg *CAO*; and 5) the concordance entry for the chant as it appears in the printed volumes of *CAO*, but following CANTUS conventions. This allows for several permutations. On entering data into the *incipit* field, the data-capture operator can instruct the GUI to provide a list of possible valid *CAO* numbers, together with their full texts. A selection from this list will result in data entries to the *incipit*, *cao*, and *chantid* fields. A major advantage of this feature of the GUI is that it speeds up the process of *incipit* and *chantid* correlation and eliminates the need to continuously consult the paper-based *CAO*. It also eliminates the keyboard entry of the *CAO* concordance data.

At this point, a short digression is called for. It should now be clear that the "Cantus Index GUI" is heavily dependent on existing data. Questions pertaining to the usefulness and accessibility of the existing data have recurred at various stages of the utility's design. Data that does not exist in a standard format that database software can readily recognize is difficult to access. The data in the CANTUS database is easy to re-use and to manipulate, precisely because CANTUS makes this information available as fixed line length ASCII files. All one needs to work with this data is the CANTUS description of the ASCII files. With this information in hand one can, for example, easily open the files in database software or design an interface with which to edit and view the

[8] http://www.uni-regensburg.de/Fakultaeten/phil_Fak_I/Musikwissenschaft/cantus/index.htm

data. If the CANTUS data had existed as text files that did not consistently follow an accepted standard, the whole process of making this data available for re-use would have been much more difficult. It would have been necessary to first convert the data into a standard format.

The difficulty of working with non-standard files in a database environment became evident during the part of this project that incorporated the electronic *CAO* published on the Cantus Planus web site. The data is available from the web site as text files. These files contain a number of irregularities as regards the number of spaces between words and units of text that, in a database table, would belong to different fields. The number of carriage returns, which would indicate the divisions into records, is also not consistent. This necessitated a rather time-consuming conversion process with Microsoft Word macros, before the incorporation of additional data into the table could begin. On the other hand, the process of adding the data with the CANTUS-style *CAO* number and concordance information was speeded up by the existence of a large amount of this data in the database-compatible CANTUS indices. The main advantage of the Cantus Planus *CAO* was, of course, the fact that its creators had already completed the tedious input phase of the complete texts of the Office chants. Even more important is the fact that they subjected their work to a thorough editing process.

An Overview of the MIDINeume ActiveX Control

The "Cantus Index GUI" is an electronic index[9] generating system designed initially to facilitate the creation of Cantus indices. It must be noted, however, that such a tool as this can very easily be modified to generate many other forms of index, limited only by the scope of data captured, and the imagination of the programmer. To this end, a desire was expressed to be able to capture more completely the melodic information of the chant, alongside the existing textual and descriptive information, for the purpose of being able to base such indices on melodic information. It is this desire that the MIDINeume ActiveX Control sets out to satisfy. Let us start by noting that computerized representations of any data must ultimately be reduced to binary codes. Standards have long been in place for the representation of text and numeric data in digital form, but the digital representation of musical notation has, arguably, evolved far more slowly. More importantly, database technology has, from its very inception, dealt transparently with representations of alphanumeric data, but

[9] We use the term 'index' somewhat uncomfortably, since it potentially conflicts with our use of the same term in the context of database technology. Wherever we use the term in this paper, the reader is to understand 'index' as understood by those familiar with CANTUS.

has never provided a data type to deal with musical data. And one could hardly expect this. What this amounts to is the following: in order to capture melodic information in a database, one would be required to translate the musical information into a series of database-compatible values, either numbers, alphabet characters, or a combination of these. One more possibility does exist: the storing of raw binary data in the database, but since this poses an inordinately high level of difficulty to the data capture operator, it should be plainly seen as a non-option. This leads us back to a manual translation of notation into numbers.

The MIDI specification[10] provides a widely accepted translation of pitch values to note numbers, along with a system for representing the placement in time of musical events, relative to a particular starting time. The underlying representation of MIDI data is binary, and may thus be stored as such along with other forms of data in a database. What is more, the MIDI specification is widely implemented in electronic keyboard instruments and such a keyboard, therefore, already translates the operator's performance into binary data (i.e. numbers). All that is required is a piece of software to capture the MIDI data from the keyboard, and to store the captured data in the nominated data field in the nominated database. Capturing such data is a primary function of a computer program known as a 'sequencer'. Unfortunately, sequencers have never needed to consider storing their data in a database, relying instead on flat files on the hard drive. What is more, sequencers typically support a much larger set of functionality than that required by this application. However, the development of sequencer software has given rise to libraries of software components written for the express purpose of capturing and replaying MIDI data. Such a library (the one employed in this project) is Paul Messick's "Maximum MIDI Toolkit".[11] Of course, the act of storing the data is only of value if that data can be retrieved again, so the software must similarly be capable of retrieving MIDI data from such a database, and presenting it to the operator in some useful form. Herein lies the prime purpose of the MIDINeume ActiveX Control: to provide the most basic sequencer functionality in a data-bound control.[12] Initially, it has been decided to present this information to the user in two forms: as a replayed performance on a MIDI instrument, and also as musical notation in a visual control. Messick's MIDI toolkit provides the core functionality for the capture, elementary manipulation, and playback of the MIDI

[10] MIDI Manufacturer's Association, *Complete MIDI 1.0 Detailed Specification v96.1*, 2nd edition, La Habra, CA: MMA, 2001. See the MMA web site for updates: http://www.midi.org

[11] P. Messick, *Maximum MIDI: Music Applications in C++*, Greenwich: Manning Publications, 1998.

[12] Microsoft Corporation, "ActiveX Controls: Using Data Binding in an ActiveX Control," in *MSDN,* October 2001 release.

data. Microsoft's data-bound control architecture provides the means whereby the data is stored in and retrieved from the database. Messick further allows us to easily expand our control to import and export standard MIDI files, the universally accepted format for the non-volatile storage of MIDI data. The final architectural component provides the visual representation of the captured data. For this we have adopted Microsoft's Foundation Classes, a widely employed framework for software development on the Microsoft Windows operating system. Within this framework, we are able to make use of János Bali's Guido font for the actual representation of the data in modernized Messine-Gothic musical notation, also referred to as the "Budapest" transcription method.[13] Since the piece of software we are writing is to be tightly integrated with the "Cantus Index GUI", we choose to package the components as an ActiveX Control, thereby providing a self contained unit of functionality that is easily integrated into the existing forms of the "Cantus Index GUI". Refer to *Figure 2* for an illustration of the preceding discussion.

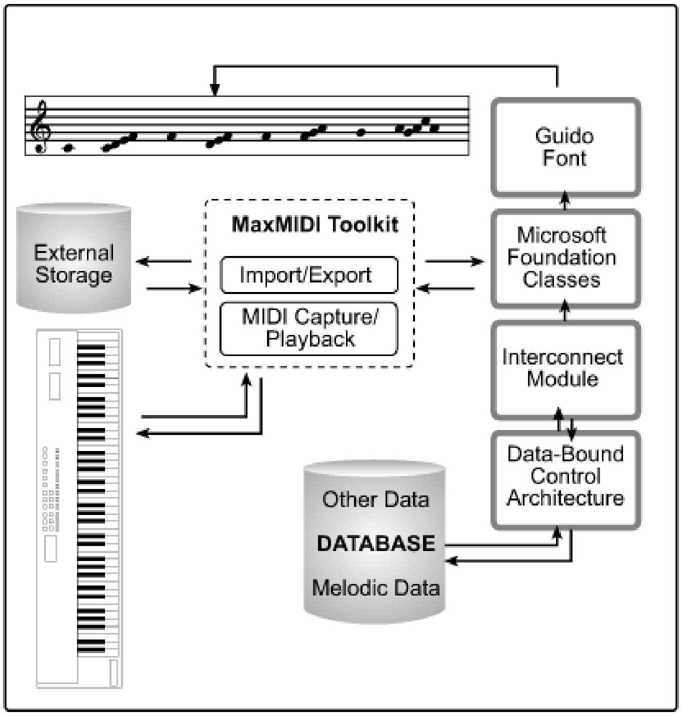


Figure 2: Architectural Overview of the MIDINeume ActiveX Control

[13] http://www.zti.hu/earlymusic/guido.htm

Having now provided an overview of the architectural options which have been employed, it might serve to digress slightly to discuss some of the problems encountered. Our first problem is the issue of quantization.[14] As long as musical data remains in the realm of notation (albeit traditional paper-based notation or digital notation), we tend to take for granted the subtle differences between the lengths (in time) of two different musical events which in fact have the same notated rhythmic value (that is to say: all quarter notes are not created equal). The same, of course, could be said of the difference in starting times between a string of like-metered note values: they will not necessarily be spaced 100% equally in performance. The desirability or not of this phenomenon is not under discussion here. It is a simple fact that, at least as far as a computer is concerned, human performances are somewhat less than perfect. Sequencers generally provide a facility to *quantize* such a performance, effectively rounding the rhythmic values of the performer to a set common denominator. In the MIDINeume ActiveX Control, the answer to this problem has, in fact, become a feature in itself. Since we have constrained ourselves to only two rhythmic values and discounted any need to have rests, we have only had need to provide for a threshold setting. Any rhythmic value shorter than the threshold is recorded as the shorter rhythmic value, and anything longer is recorded as the longer value. A pleasant side effect of this simple algorithm is that the chant may now be captured melisma by melisma, with the performer being free to pause indefinitely between melismas to contemplate the interpretation of the next melisma. The result is less a record of the performance, but provides an easier data-entry tool which, after all, is what we have set out to do.

The second issue deserving mention is that of polyphony. The MIDI specification is designed to record polyphonic performances, and does so through a simple mechanism: it separates the start of a note from its end, recording both as distinct musical events joined by a common note number. This allows multiple notes to be sounded and ended at independent points in time. On an aside, one might notice that this mechanism does not permit a note to sound before its previous sounding has ended (a purely academic point, perhaps, and not really a practical consideration). What does pose a problem, however, is that legato playing, in fact, constitutes polyphony. Each note's ending time overlaps the next note's starting time. We are clearly interested in monophonic lines here, and our solution is brutal and effective. We disregard all "end of note" events received from the instrument, opting instead to interpolate our own, placed directly before the next "start of note" event. Again, the result is less a record

[14] Quantization: a term which has a very specific meaning to those well acquainted with MIDI, but the general usage (in Physics etc.) will not mislead the reader in understanding its usage here.

of the exact performance, and more a representation of the musical data being examined, regardless of whether the lines are played legato or otherwise.

We turn now to the graphical representation of the captured data. It was decided early in the project to model the notation on that employed by László Dobszay. One approach would have had us render this notation by plotting geometric shapes on the visible surface of the control, but we favoured a more economical approach (arguably a more elegant, certainly a more common one), that of employing a notation font. At this point, it was proposed that we design our own, but this was ultimately made unnecessary by the kind cooperation of László Dobszay, who provided us with a copy of the 'Guido' font for this purpose. At present our control implements only the most basic functionality of this tool (for example: no accidentals, no stems or connecting strokes), with efforts underway to completely integrate the full set of functionality into the next revision.

We move now to the actual representation of the data in the database. As a point of departure, we set about storing the MIDI data (essentially a stream of binary numbers) as a Binary Large Object (BLOB) in the assigned database field. It soon became clear that this strategy would later restrict the researcher's ability to perform advanced data searching and sorting operations, since (as noted previously) the manipulation of binary data in a database is less than trivial. This would have led to a situation in which the MIDINeume ActiveX Control would have had to include functionality for every conceivable type of query or operation on the plainchant data type, and we suspect that this would soon have become a serious bottleneck for many researchers. The binary storage format is still in use in the current version of the control, but we are now pushing toward the adoption of a more open standard employing an alphanumeric representation of the data. We note with great enthusiasm the work of Louis Barton in providing an XML specification for the encoding of plainchant notation.[15] We feel, however, that employing Barton's method at this stage is superfluous to our stated goal. We are concerned here with the transcribed melodic data, not the exact representation of the neume on the manuscript. The intent is to populate the database with data that can be categorized, sorted, and searched for patterns and anomalies. As stated above, databases are equipped to provide these functions where numbers and alphabetic characters are employed. Performing the same operations on binary data is far more problematic. We thus opt to store the captured data in the database field as a series of alphanumeric characters. This immediately opens up a vast sea of data-related operations, all native to the database engine, and implemented far

[15] http://scribe.eas.harvard.edu/barton/semantic.htm

more easily by the creative researcher. Again, we have found a model as point of departure, this time in Leland Smith's 'Score' data entry language, but have found it necessary to adapt and simplify that model somewhat.

We felt it imperative that the MIDINeume ActiveX Control be able to share its data with other MIDI applications through an "import/export" functionality employing a standard file format. The MIDI specification defines a .mid file format expressly for this purpose. The contents of such a file, however, extends some way beyond our requirements. We have thus opted to make use of Paul Messick's function library, but ignore much of the data contained in a midi file (in the case of import), replacing said data when a file needs to be created once more (in the case of export). Specifically, we discard all metadata, as well as all data on tracks other than track 1. Of the data on track 1, we further discard all but the "note on" messages (refer to our earlier discussion of the treatment of "note off" messages). As a result, importing a file into the control and then exporting it again will, in most cases, not yield the same file. Again, we are satisfied that this strategy meets our specific needs. We will certainly expand the import facility at some point in the future to allow data extraction from tracks beyond track 1.

Looking toward some of the design compromises: it is of some concern that the captured MIDI data potentially represents a large amount of redundant data (much of the same melodic material will occur in many different chants). We do envisage a truly normalized structure at some point in the future, wherein the database will only store one instance of any given melodic fragment, and the chant's record will contain pointers to each of the fragments making up the chant. This, of course, will make new categorizations plainly apparent, and will provide far more analytical power, but needs more thought in order to assess the feasibility of implementing such a design. It could, if poorly implemented, effectively hinder certain kinds of analyses.

On a further design compromise, it was decided to not concern ourselves, for the moment at least, with the capture of liquescence. This has not been done without consideration for the incorporation of this feature at some point in the future. It was initially proposed that MIDI control change messages be adopted to represent the point in time at which a certain ornamental neume is found. Of course, no attempt is made to interpret the melodic implication of the ornamentation. We are simply concerned with alerting the user that such ornamentation appears in the original manuscript. The use of control change messages does serve this purpose, but we are concerned about violating the integrity of the design and purpose of the MIDI specification. We therefore look now toward employing something more suited to this purpose: perhaps a non-registered parameter number, or more likely a meta-event of some kind. Finding a suitable solution to this problem will likely be a prime directive in the next revision of the control.

The "Cantus Index GUI" has given us the opportunity to apply ourselves to the most fundamental problems, and to establish the basic architecture. What we have, in essence, is a component that can easily be integrated into software written in a wide range of languages, integrated into documents, and even embedded in web pages. We would like to see this as part of a larger development drive toward a complete, modular, database-aware, polyphonic sequencer with strong support for notation and analysis. We feel that the educational and research value of such a tool would easily justify the effort, especially if it were designed to accommodate a range of specialized needs. It will, of course, move somewhat outside of the boundaries of plainchant research in this expanded form, but we do intend to retain complete backward compatibility, and to ensure that each revision is completely able to satisfy this first application. Ultimately, we feel that this tool may even evolve to contribute at some level to composition and performance practice.

Conclusion

The "Cantus Index GUI" seeks to accelerate the capture of data describing plainchant sources, and to encourage standardization with a view to future consolidation and analyses of this data. It has further brought its authors to confront many unique issues as regards the nature of the pertinent data, its structure, efficient storage, and the flexibility with which it can be manipulated. The addition of the MIDINeume ActiveX Control points the way to some of the expansion possibilities. We hope to engage the plainchant research community at large in further developing this utility so that it might truly address the problems confronting plainchant research.