MacSet [4] is a computer application intended to assist in learning and teaching basic pitch class set theory and analysis of post-tonal music. It is an umbrella application under which several pitch class set (pcset) theory-related tools are collected. It is designed and programmed by Marcus Castrén, Mikael Laurson and Mika Kuuskankare at the Sibelius Academy, Finland. The project’s Web site can be located at www.siba.fi/MacSet (pending public release). The Web site provides some background information about MacSet and also allows to download the program for free. This paper introduces the MacSet version 4, a stand-alone application for Mac OS X, Windows, and Linux.

1. INTRODUCTION

Theorists have introduced a number of computer programs focusing on applications of pcset theory. CMAP (Castine), SetMaker/pSetMaker (Buchler) and WinSIMS (Isaacson) are typical examples of applications that provide an assortment of observations presented mostly in a textual form. athenaCL [2], in turn, is an open-source composition tool that, among other things, provides a dedicated library for inspecting, searching, and visualizing the properties of set-classes, comparing and searching similarity measures. Yet another, albeit more specialized approach, is provided by Lisp-based Counterpoint Assistant [10]. Furthermore, Humdrum [9] provides a collection of tools specialized for serial analysis and both PWGL [12] and OpenMusic [3] provide libraries computer-aided music theory, analysis and composition (see, for example, OpenMusic’s MathTools [1]).

MacSet 4 is the result of recent efforts to port our pcset theoretical software to multiple platforms. The previous versions, dating back more than two decades, were developed exclusively for the Macintosh Operating System using the Macintosh Common Lisp. MacSet 4 is programmed with LispWorks Common Lisp (www.lispworks.com) and its CAPI (Common Application Programmers Interface) library for implementing the user interface (UI). Currently, MacSet 4 runs on three platforms: Macintosh OS X (Universal binaries), Windows XP, and Linux (Ubuntu 6.10 and SUSE 10.1 distributions).

The following list enumerates some of the new features found in MacSet 4: (1) MacSet is modular and built out of a set of specialized UI components designed particularly for illustrating pcset-theoretical information; (2) there is a new module interface that makes MacSet extensible by using Lisp; (3) the data is interchangeable across platforms; (4) a dedicated tool can be used to create interactive content (e.g., presentations, demos, tutorials, teaching material) in a modular fashion; (5) by using the application preferences it is possible to adjust the appearance and behavior of MacSet to suit a particular user; and (6) the functionality and features of MacSet 4 are identical across all three platforms.

The rest of the paper is organized as follows. First, we give an overview of the MacSet graphical user interface (GUI). Then we present the default MacSet Workspace consisting of three built-in modules and discuss the new music notational capabilities of MacSet 4. In Section “MacSet Modules” we introduce the new interface that allows to create user-definable extensions (modules). The paper ends with some concluding remarks.

2. THE MACSET GUI

The MacSet UI is completely rewritten from the ground up to create a modular, object-oriented UI that is based on direct manipulation [5]. Furthermore, MacSet 4 incorporates several enhancements, both in internal architecture and UI, that make it easier and more intuitive to use, allow extensibility, and make it more modular and open-ended.

An important part of the rewriting process has been the reprogramming of the individual UI components. The functionality has been separated from the set theoretical code. The new UI components work as any button or widget provided by the operating system: they are reusable and in most cases also context free. They are are also dynamic, i.e., they adjust their size and content automatically. Furthermore, the functionality of the UI components has been streamlined and unified. Now, every component has a dedicated context sensitive menu, etc. There are also some new ways to interact with the components such as an on-screen keyboard entry. The UI components have undergone also some visual enhancements. A lighter touch in terms of color is applied (no more black and white UI components) and majority of the components now utilize transparent colors.

MacSet GUI is based on several important concepts. First, MacSet is a multi-window system where the user
can create windows out of provided prototypes. Each window, in turn, has its own individual state which can be saved. There is no limit of how many windows the user can create. Second, MacSet windows are event-driven. The user interacts with the current window using various input devices such as mouse, computer keyboard or MIDI keyboard. Third, MacSet, tries to use optimal representation for input and output. For instance, when entering set-class names, pitches or chords, the system provides visual tools such as pop-up menus or on-screen piano keyboards. For output MacSet uses besides textual format various dynamic graphical objects, such as clock-face representation, matrices and music notation. Fourth, graphical objects can be manipulated directly. If the state of an object is modified then all related objects are also modified accordingly. Thus the state of an active window is always up to date.

3. THE MACSET WORKSPACE

MacSet is built out of modules. There are three built-in modules that comprise the default MacSet Workspace. The modules are Sets, Inclusions, and Chords. There is no specific order or hierarchy between them. Each one is specialized for a certain task. The current scheme also allows the user to extend MacSet by defining his/her own modules which can coexist alongside with the built-in ones reusing the UI components of MacSet.

The Sets Module (Figure 1) is for basic operations on pcsets. There are two input methods. The user plays an on-screen keyboard, or clicks into a pop-up menu of set-class names. Pcsets are represented visually with the help of a clock-face: small circles or hours on its perimeter represent pitch-classes. A set can be transposed or inverted with a click of the mouse. Among other immediately observable aspects are types of symmetry, successive-interval arrays, interval-class vectors, set-class names, etc.

In the Inclusions Module (Figure 2) the user selects a referential pcset and some cardinality-classes (1-12). All pcsets of cardinalities that realize the inclusion relation with the referential set will then be returned.

The Chords Module (Figure 3) allows the user to examine the set-class identities of chords and/or melodic lines, as well as study their intervallic properties (interval successions, interval-class and mod-12 interval vectors, figured bass structures, etc). With it the user can also prepare analyses that can be saved to a file.

4. ENHANCED MUSIC NOTATION

The music notational front-end of MacSet 4 has been substantially enhanced and augmented when compared to the previous versions.

Currently, the Music Notation Pane (MN) can be used to display musical material in several different ways: as individual chords, chord sequences, and as piano-roll notation. Figure 4 gives an example of the MN displaying a fragment of a J.S. Bach chorale written in the piano roll notation. The Pitch information inside this panel is editable. The notes can be transposed, added and deleted using the mouse and keyboard.

There is also now an annotation system that provides a collection of predefined annotations and also allows the user to create new ones. The annotations can be used to display analytical information directly in the score attached to the relevant musical objects.

Furthermore, it is possible to play the musical material entered in the MN. MacSet uses MidiShare [7] for playback.
4.1. File I/O

The native format of the MN is a list of executable Lisp objects. This format also retains the annotation information attached to chords (see Subsection 4.2).

There are currently several ways to export data from the MN. The following list enumerates the current set of supported file formats: ASCII (Lisp), LilyPond [13], MIDI, and MusicXML [8]. It is also possible to import MIDI files (MacSet utilizes the AllegroConvert [6] utility for MIDI file I/O).

4.2. Annotations

Through a simple protocol the user can create different kinds of annotations to be displayed as a part of the music notation. The annotations are positioned automatically according to a predefined scheme. In the following we examine the system through two short examples.

First, we create an annotation that displays the harmonic interval between two consecutive notes. The following gives the complete definition:

```lisp
```

The `datum`-method (above) calculates the intervals between two consecutive notes inside a chord. Here, the `:placement` attribute is set to :right-between-notes. The effect can be seen in Figure 5. The interval information is placed horizontally on the right side of the chords and vertically between the corresponding notes. The information is also calculated dynamically. Thus, in case the user changes the pitch material the annotations update automatically the displayed information and positioning. Note, that the intervals are arranged in alternating columns in order to avoid collision between the pieces of information.

Furthermore, the use of the :add-tool attribute allows us to quickly add a rudimentary UI component (check-box) in the Music Notation Pane. The check-box can be used to turn the annotation display on or off.

Our next example is relatively more complex as it requires some Lisp programming. Here we define a dynamic contour break-point function that is displayed above the staff lines. Figure 6 shows how the annotation looks in a score and Figure 7, in turn, gives the annotation definition in Lisp.

As can be seen in the previous examples there is actually very little that needs to be defined in order to display quite elaborate information along with the music notation. The various side effects provided by this system make it quite powerful. It is also important to note that all the annotations are able to adapt themselves to the different notation styles in MacSet.

5. THE MACSET MODULES

Finally, we give an example of a user-defined module that implements a chord multiplication application in the spirit of Pierre Boulez [11]. Due to the space limitations the code that produced this example cannot be shown here. Suffice to say that it consists of 46 lines of Lisp and takes advantage of the UI component scheme of MacSet.

Our new module consists of two separate views (see Figure 8). The top view gives the pitch material (i.e. series) that in this case comprises of four sets that together fill the chroma. The bottom view gives the resulting multiplication matrix consisting of 16 chords. Using the built-in functionality of MacSet the user can edit the series by transposing or adding and removing notes. Each operation is visualized directly in the multiplication matrix as MacSet updates the result dynamically when the user edits the pitch material.

6. CONCLUSIONS

This paper presents MacSet, a free visual cross-platform computer application aimed at pitch-class set theoretical applications of music. MacSet runs on several platforms including Macintosh OS X (both Intel and PowerPC), Windows XP, and Linux.

MacSet can be used as an assistant in learning and teaching basic pitch class set theory and in analysis of post-tonal music. It can also be used to produce presentations, demos, tutorials, and teaching material.

MacSet is modular and can be extended by the user. The data produced by MacSet is interchangeable across all the supported platforms. Currently, MacSet provides...
extensive music notational facilities where analytical in-
formation can be viewed and visualized in many different
ways.

7. ACKNOWLEDGMENTS

The work of Mikael Laurson and Mika Kuuskankare have
been supported by the Academy of Finland (SA 105557
and SA 114116).

8. REFERENCES

[1] Andreatta, M., C. Agon Amado, T. Noll, and
E. Amiot. “Towards pedagogability of mathemat-
ical music theory: Algebraic models and tiling prob-
lems in computer-aided composition”, Proceedings
of Bridges 2006, Mathematical Connections in Art,

University, New York, NY, USA, 2005.

[3] Assayag, G., C. Rueda, M. Laurson, C. Agon, and
O. Delerue. “Computer Assisted Composition at IR-
CAM: From PatchWork to OpenMusic”, Computer

“MacSet: A Pceset-Theoretical Computer Applica-
tion”, 26th Annual Meeting of the Society for Music
Theory, (Madison, USA), 2003.

[5] Cooper, A. About Face. The Essentials of User In-


the open source softwares”, Proceedings of the Inter-
national Computer Music Conference (ICMA, ed.),
pp. 311–313, 1999.

File Interchange”, Third International Conference
on WEB Delivering of Music, (Los Alamitos, CA),

[9] Huron, D. “Music information processing using the
humdrum toolkit: Concepts, examples, and lessons”,
Computer Music Journal, vol. 26, no. 2, pp. 15–30,
2002.

for atonal counterpoint”, Computer Music Journal,

Boulez: Analysis of Le marteau sans maître. PhD

PWGL”, International Computer Music Conference,

a system for automated music engraving”, XIV Col-
loquium on Musical Informatics (XIV CIM 2003),
(Firenze, Italy), 2003.