A Visual Workspace for Polyphonic Texture-Writing

Owen Cannon

New York University, owen.cannon@nyu.edu

*Abstract* – The practice of species counterpoint figures prominently in music composition/theory, music production, and in screen music as a practical introduction to musical precepts of the western classical tradition. This paper will serve as a run-down of software artifacts used in the music-conversion package for the OpenMusic visual programming environment to support the visual species polyphony canvas.

*Index Terms* – Polyphony, OpenMusic, Visual Programming, CSP, Logic Programming

# ‘Gradus ad Parnassum’

From the [*List of Works entitled ‘Gradus ad Parnassum’*](https://en.wikipedia.org/wiki/Gradus_ad_Parnassum) on Wikipedia:

The Latin phrase gradus ad Parnassum "steps to Parnassus". It is sometimes shortened to gradus. The name Parnassus was used to denote the loftiest part of a mountain range in central Greece, a few miles north of Delphi, of which the two summits, in Classical times, were called Tithorea and Lycoreia. In Greek mythology, one of the peaks was sacred to Apollo and the nine Muses, the inspiring deities of the arts, and the other to Dionysus.[1]The phrase has often been used to refer to various books of instruction, or guides, in which gradual progress in literature, language instruction, music, or the arts in general, is sought.

The jjf-species1 Visual Program

As shown in Figure 1, the main patch window contains each of the three phases outlined above. The make-screamer-variables subpatch accepts a list of midi-notes corresponding to the notes of the cantus firmus and generates variables. Each variable represents a set of values corresponding to MIDI note values, its finite domain, which in this case is the set of integers from 55 to 84. These values represent the pitch range from the G below middle C to the C two octaves above middle C. The variables are sent to the subpatch labelled level2 (see Figure 2), containing objects representing the rule set being applied in the search procedure. Each rule represents a constraint on the input variables corresponding to rules governing voice motion in species counterpoint. As they are applied to their input, they restrict the domain of possible values of the input variables corresponding to the main theme and the second voice. Finally, the solver represented in the main window with an icon labelled om-solutions steps through the domain of the input that has been passed through the level2 subpatch. As it fixes each input variable the solver triggers the system to reevalute the rules against the other variables, until it arrives at a combination the meets all conditions. When all possible combinations have been exhausted, and the patch formats the results so they can be viewed in a display window or exported into various file formats.

# Nested list structure for representing note-to-note motion

The generated variables, combined with the list corresponding to the main theme, comprise a nested list structure that is referred to as a *sequence* throughout the music-conversion environment. Each list at the top level of a sequence corresponds to a voice in a polyphonic texture as shown in Figure 1.

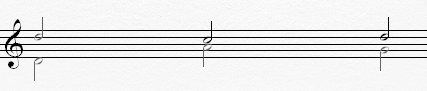


Figure 1: Notated representation of the list structure

((74 72 74 … )

(62 69 67 … ))

Second species note-to-note motion can be represented using nested lists as shown in Figure 2.



Figure 2: Notated representation of the list structure

((74 (75 72) 74 … )

(62 69 67 …))

The *mat-trans* or matrix-transpose function reorients these nested list structures to give a list of all of the simultaneities in a multivoice-texture. For example, the second-species seuqence

((74 (75 72) 74 … )

(62 69 67 … ))

appears as follows after undergoing matrix-transpose:

(mat-trans

(flatten-seqc

‘((74 (75 72) 74 … )

(62 69 67 … ))

((62 74)

(69 75)

(69 72)

(67 74)

…

In this way, rules designed to apply to first-species counterpoint can be applied to complex list structures representing voices at the fifth-species.

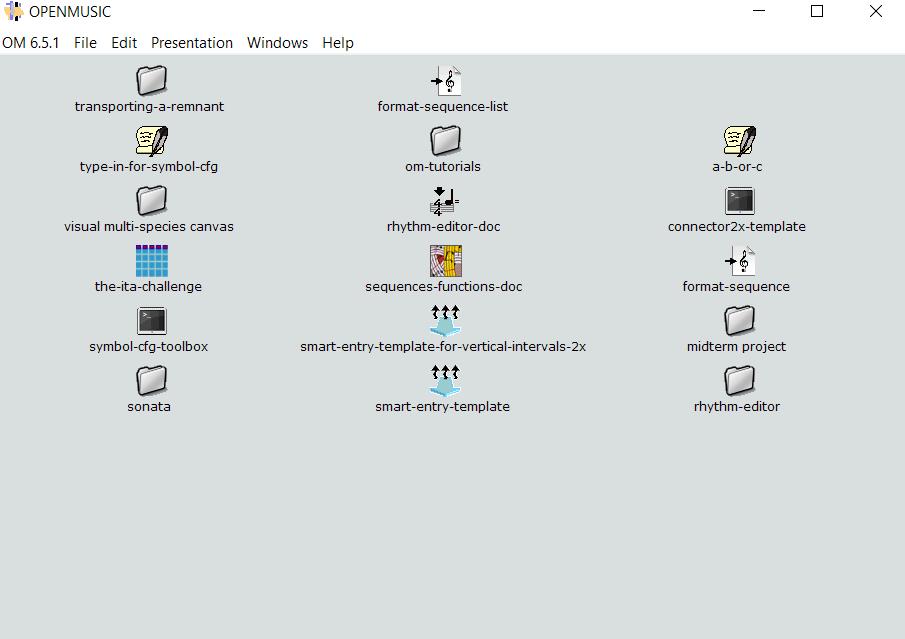


Figure 3: The main program window.

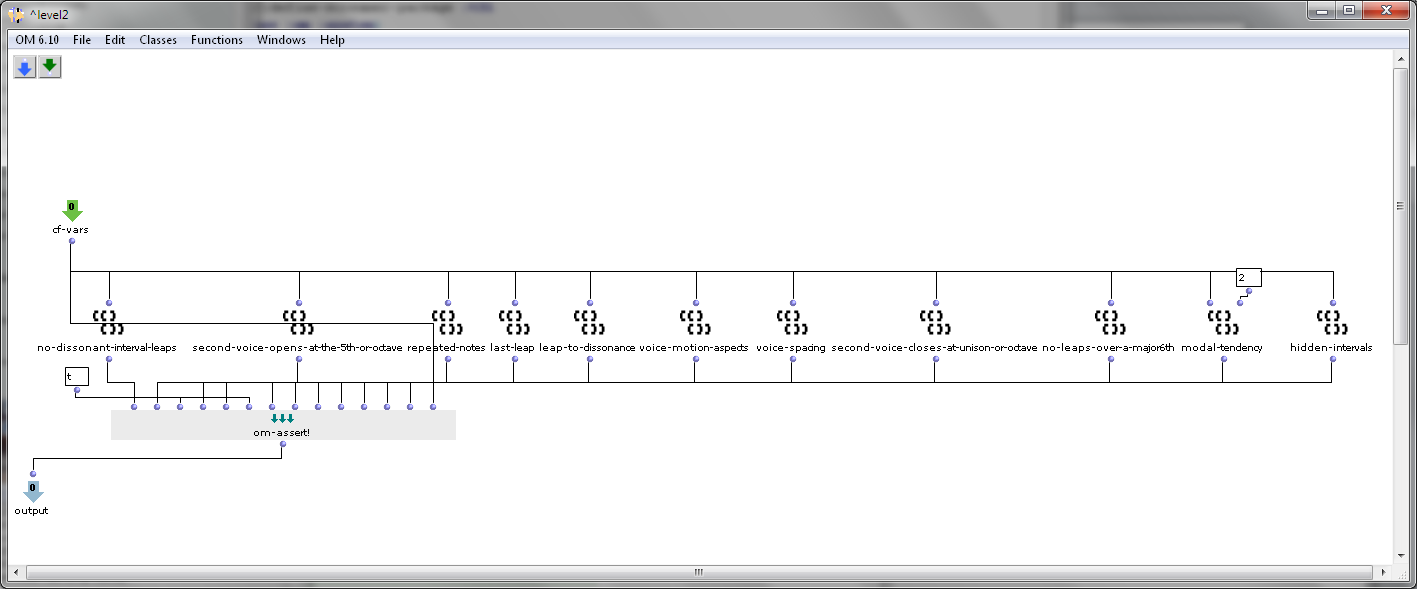


Figure 4: Patch window for applying rules to nondeterministic variables corresponding to the notes of the cantus-firmus and counterpoint. The om-assert! object applies the rules defined in each folder.

# Limitations of the CSP Approach

The major limitation with working with boolean constraints in connection to music is their all-or-nothing nature. Most of the precepts governing species counterpoint are not hard and fast rules. They are guidelines intended to be weighed against their overriding aesthetic principle, the sense of the mediated individuality of voices within the resulting texture.

One possible means of addressing this problem would be to make use of the solution-scoring functionality built into Screamer. Screamer includes a Best-Value function that assess a variable that in effect weighs any given solution against some user-defined criteria. If, for instance, the point system corresponded to the number of parallel consonant intervals times -1, the Best-Value function would favor solutions with the least number of parallel consonant intervals. A supplementary set of rules could be designed to return numeric values that could be weighted and summed together to comprise a scoring system.

# Functional Summary of Visual Programs and Species Counterpoint Precepts

This section includes a summary of species counterpoint rules as given in the Wikipedia entry for Counterpoint along with the names of patches from the motion-rules folder that implement the rules. The rules objects included in the *jjf-utilities* folder do not fully reflect the rules governing voice motion in species counterpoint as they appear in *Gradus ad Parnassum*. They give a rough approximation of the basic rules of species counterpoint that is intended to convey the technological possibilities of logic-programming.

1. *The* [*final*](https://en.wikipedia.org/wiki/Mode_%28music%29#Western_Church) *must be approached by* [*step*](https://en.wikipedia.org/wiki/Steps_and_skips)*. If the final is approached from below, then the* [*leading tone*](https://en.wikipedia.org/wiki/Leading_tone) *must be raised in a minor key (Dorian, Hypodorian, Aeolian, Hypoaeolian), but not in Phrygian or Hypophrygian mode. Thus, in the Dorian mode on D, a C♯ is necessary at the* [*cadence*](https://en.wikipedia.org/wiki/Cadence_%28music%29)*.*[*[5]*](https://en.wikipedia.org/wiki/Counterpoint#cite_note-5)

The **last-leap** rule applies the counterpoint rule regarding cadence patterns allowed in the second voice.

1. *Permitted melodic intervals are the perfect fourth, fifth, and octave, as well as the major and minor second, major and minor third, and ascending minor sixth. The ascending minor sixth must be immediately followed by motion downwards.*

The **accepted interval leaps** rule filters each voice according to a list of permitted melodic intervals. The **leaps-return-to-a-different-note** applies the constraint that for every three-note sequence containing a leap the return note can’t be the same note as the initial note.

1. *If writing two* [*skips*](https://en.wikipedia.org/wiki/Steps_and_skips) *in the same direction—something that must be only rarely done—the second must be smaller than the first, and the interval between the first and the third note may not be dissonant. The three notes should be from the same triad; if this is impossible, they should not outline more than one octave. In general, do not write more than two skips in the same direction.*

The **two-leaps-in-the-same-direction** rule applies the rule that for each three note group containing two leaps in the same direction, the first leap is smaller than the second and they not outline a dissonance. This object does not process triads, although there is a set-equality function in the library that could be used to check unordered groups of variables against pitch-class invariant sets (in this case the pitch class set 0 3 7). Because this rule has to be applied to every permutation of its input in order to achieve pitch-class invariance, its easy to go over the stack size limit (i.e. the memory-related limit on the number of successive function calls being processed at a given moment). Users with a license for *Lispworks Pro* would have access to raise the stack size restriction before compiling Openmusic themselves.

1. *If writing a skip in one direction, it is best to proceed after the skip with motion in the other direction.*

The **voice-motion-aspects\_2** rule applies the constraint that leaps are to be followed by a step in the opposite direction.

1. *The interval of a* [*tritone*](https://en.wikipedia.org/wiki/Tritone) *in three notes should be avoided (for example, an ascending melodic motion F–A–B♮)[*[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)*] as is the interval of a seventh in three notes.*

The **tritone-over-three-notes** rule applies the rule that the first and last note of each successive group of three notes cannot be a tritone where all three notes in the group are moving in the same direction.

1. *There must be a climax or high point in the line countering the* [*cantus firmus*](https://en.wikipedia.org/wiki/Cantus_firmus)*. This usually occurs somewhere in the middle of exercise and must occur on a strong beat.*

The **climax-point** rule applies the constraint that one of the notes between the first and the last notes has to be higher than the rest of the notes in the voice.

1. *An outlining of a seventh is avoided within a single line moving in the same direction.*

## Summary of Main Rules for Species Counterpoint

1. The counterpoint must begin and end on a perfect [consonance](https://en.wikipedia.org/wiki/Consonance_and_dissonance).
2. [Contrary motion](https://en.wikipedia.org/wiki/Contrary_motion) should predominate.
3. Perfect consonances must be approached by oblique or contrary motion.
4. Imperfect consonances may be approached by any type of motion.
5. The interval of a tenth should not be exceeded between two adjacent parts unless by necessity.

*hidden-intervals:* applies conditions for avoidance of parallel motion.

*modality:* applies the condition that all input variables modulo 12 have to belong to a pitch class set corresponding to the modality of the cantus firmus. The mode is d-dorian in these examples.

*last-leap* applies the condition that the last note of the counterpoint proceed from either the step above the tonic or the natural or sharp leading tone.

*leap-to-dissonance* applies the condition that the between any two consecutive simultaneities neither voice can move further than a step where there is motion to a dissonant interval.

*modal-tendency* applies the condition that tones resolve according to their modal tendency.

*no-dissonant-interval-leaps*

*repeated-notes* allows up to one repeated note in the second voice. This is due to the fact that the first solution the solver arrives at is typically a series of repeated notes.

*second-voice-closes-at-unison-or-octave* determines cadance patterns for the second voice

*second-voice-opens-at-the-5th-or-octave* applies the condition that the second voice proceeds from the octave or perfect 5th above the first note of the C.F.

*voice-motion-aspects* is intended to apply rules pertaining to parallel intervals between the C.F. and the second voice. At this point it bans parallel 5ths and does not address the other intervals.

*voice-spacing* applies the condition that every note of the second voice be higher than the corresponding note in the C.F.

# Other Resources for Logic Programming Environments for Music Composition

I did not experience a lot of success finding comparable software environments applying the CSP approach to music composition related problems. One lead was the *CounterpointService* web application. Unlike the Visual Workspace for Polyphonic Texture Writing, individual voice-motion rules cannot be turned on and off. The facility for turning rules on and off could be helpful for composers interested in multi-voice textures that may not fully adhere to the precepts of species polyphony, musical textures like ‘planing’, for example.

# Conclusion

With Algorithmic composition environments and logic-programming software it becomes possible to automatically resolve difficult harmonic constructs endemic to musical polyphony. The availability of tools that circumvent the traditional pathways to compositional mastery could alter disciplinary boundaries within institutions that have traditionally administered knowledge of this craft. The possibility that technology could one day enable composers to successfully apply a copy-and-paste approach to creating music whose complexity rivals that of the sacred music of JS Bach and his contemporaries would raise many questions. There is a long way to go before music systems such as this one can comprehensively describe musical structures endemic to western classical music of this era. The limited success I have achieved here may indicate the possibilities afforded by this technology.

# References

*CounterpointService.* (2018, 10 01). Retrieved from Algorithmic Counterpoint Composition: https://counterpointservice.cfapps.io/

Silva, N. (n.d.). *Reference Page.* Retrieved from Screamer: An Environment for Logic Programming: http://nikodemus.github.io/screamer/

Wikipedia Contributors. (2018, October 17). *List of Works Titled Gradus ad Parnassum.* Retrieved from Wikipedia: https://en.wikipedia.org/w/index.php?title=Gradus\_ad\_Parnassum&oldid=864406045

Author Information

**Owen Cannon,** M.M. Candidate, Music Performance and Theory-Composition, New York University.