GenSession: a Flexible Zoomable User Interface for Melody Generation

François Cabrol[1], Michael J. McGuffin[1], Marlon Schumacher[2], and Marcelo M. Wanderley[3]

[1] École de technologie supérieure, Montréal, Canada,
francois.cabrol@live.fr, michael.mcguffin@etsmtl.ca
[2] IDMIL - DCS - CIRMMT, McGill University
marlon.schumacher@music.mcgill.ca
[3] IDMIL - CIRMMT, McGill University
marcelo.wanderley@mcgill.ca

Abstract. GenSession is a zoomable user interface in which short clips of musical passages can be created and positioned on a 2-dimensional workspace. Clips can be created by hand, or with automatic generation algorithms, and can be subsequently edited or sequenced together. Links between clips visualize the history of how they were created. The zoomable user interface is enhanced with an automatic re-framing mode, and the generation algorithms used support dynamic parameters that can be sketched as curves over time. GenSession allows melodies and sequences of chords to be generated quickly without expert knowledge. Initial user feedback is reported.

Keywords: zoomable user interface, ZUI, music generation, sketching

1 Introduction

Computer applications for music composition can be positioned on a spectrum of ease-of-use. At one extreme are very easy-to-use tools, aimed at a large population of novice or casual users. These applications typically hide details of their implementation and provide functionalities for producing musical output in a specific musical style or genre. Examples include PG Music's Band-in-a-Box and SongSmith [14]. At the other extreme we find music programming environments and domain-specific-languages, which provide greater flexibility, however in turn may require extensive training and expertise before using, for example SuperCollider [11], athenaCL [3], or visual programming environments such as Patchwork, OpenMusic [4] and PWGL [9]. Toward the middle of this spectrum are tools focusing on a subset of possibilities, typically exposing specific compositional parameters through graphical user interfaces. These applications aim at a balance of flexibility and ease-of-use and require little or moderate training.

We propose a novel interaction style that is appropriate for tools in the middle of this ease-of-use spectrum, and we demonstrate this style in a software prototype called GenSession, a zoomable user interface for melody generation.

GenSession allows a user to generate short segments of melodies (or sequences of chords), called clips. Clips may be freely positioned within a 2D workspace, allowing the user to group or position them according to any criteria, similar to positioning icons on a virtual desktop. Multiple clips with different generation parameters may be created. Users may thus review multiple alternatives before selecting among them. Users may also combine or mix the content of different clips, or use generation algorithms to create new variants based on existing clips. Furthermore, a network of relationships between the set of clips is displayed in the form of graph edges, enabling the user to see how each clip was generated, and allowing the user to visualize and retrace the history of their creative process (Figure 1).

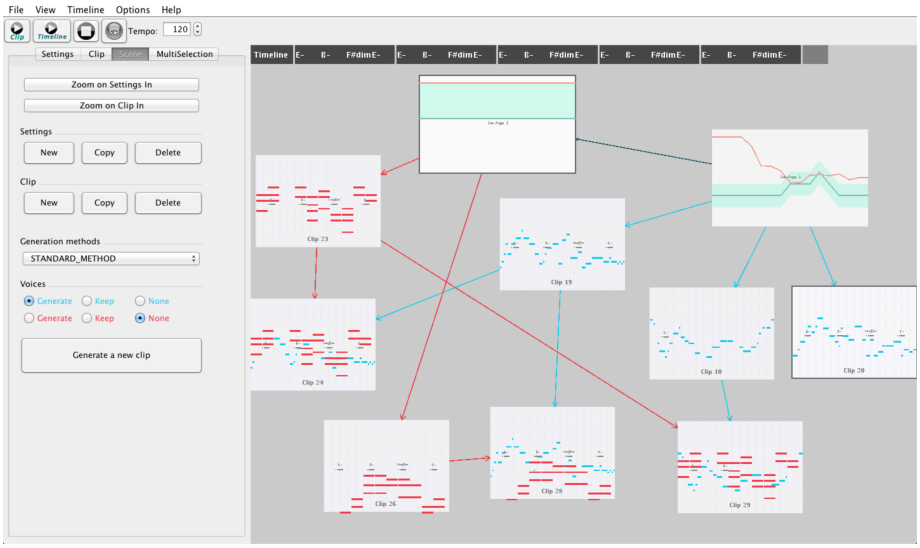

Fig. 1. The main window of the prototype. The 2D workspace displays a network of interrelated settings objects and clips.

GenSession also leverages concepts from Zoomable User Interfaces (ZUIs) [5], allowing the user to zoom into a single clip (to see a piano-roll type view of the clip) or zoom out to an overview of all clips. The user may quickly toggle between these two views with a single keystroke, during which the zooming is animated with a quick and smooth visual transition, making it easier for the user to understand the relationship between the two levels of abstraction. Furthermore, when the user is zoomed out and is dragging clips to reposition them on the 2D workspace, our prototype supports "automatic re-framing", with the main window automatically panning and zooming to maintain the visibility of all clips at any time.

GenSession allows a variety of parameters to be chosen by the user and fed into music generation algorithms. Parameters may be boolean, integer, floating point, or even quantities that vary over time, which we call dynamic parameters. Such dynamic parameters may be drawn as curves with the user's mouse (Figure 2), allowing the user to informally sketch out how different generation parameters (such as note duration) should vary over the course of a clip.

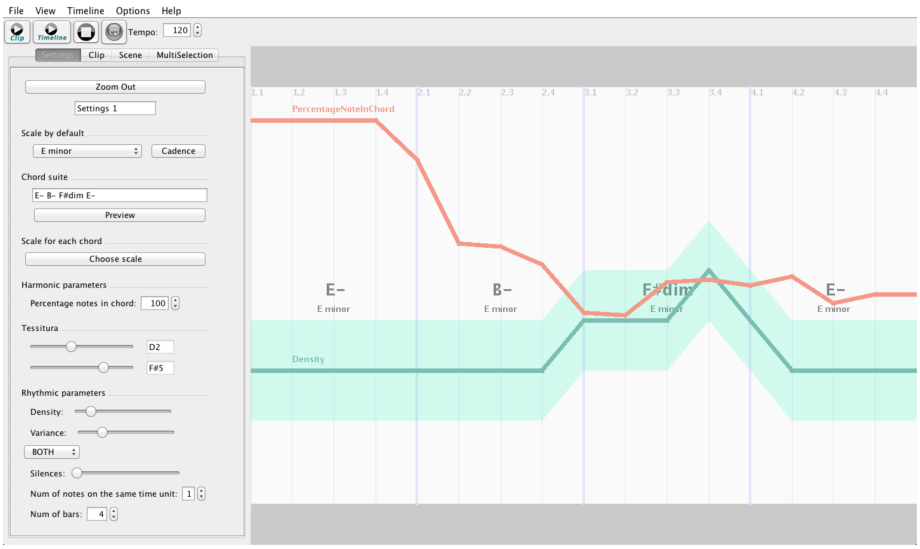

Fig. 2. The user has zoomed in on a settings object, to see and edit its properties. This settings object generates clips with 4 bars, with target chords E-, B-, F♯dim, E-, respectively. Dynamic parameters appear as curves that can be sketched with the mouse. The red curve is the percentage of generated notes that should fall on notes of the target chords, and the green curve corresponds to rhythmic density. The semi-transparent envelope around the green curve corresponds to the allowed variance in rhythmic density.

The result is an interactive music generation tool which allows musicians without a computer music background to quickly generate new melodies and variants of melodies. These melodies can be reviewed, modified, and recombined with an easy-to-use graphical user interface that helps the user keep track of and visualize their history and relationships between clips. Our contributions are (1) the use of a zoomable user interface, with optional automatic re-framing, for navigating a graph of relationships between clips, (2) the ability to sketch out curves to define dynamic parameters for generation, (3) initial user feedback from music students and researchers that partially confirm the value of our prototype's features.

2 GenSession Prototype

GenSession is a Java application that uses the `javax.sound` API.

2.1 General Overview

GenSession allows the user to generate short passages of music, called *clips*, using different generation settings, and then listen to them, reposition them on a 2D workspace, copy them, modify them, and delete those that are no longer desired. In the main window (Figure 1), a panel of options appears on the left, and most of the main window is taken up by the *scene* (workspace), which contains two kinds of nodes: clips, and *settings objects* (used to generate clips). Nodes are connected by arrows to indicate which nodes were used to generate other nodes. These arrows help the user recall the history of operations that were performed.

With the mouse, the user may freely pan and zoom within the 2D scene, or activate an automatic re-framing mode whereby moving a clip causes the scene to automatically pan and/or zoom to maintain the set of all clips centered and visible. The user may also select a single node and modify it through the panel of options on the left, or zoom in on the node to see and modify details of its settings or content. Hitting a key on the keyboard allows the user to rapidly zoom in or out of the selected node, allowing the user to switch between a global overview and a focused view of a single node. These keyboard-driven transitions between "zoomed out" and "zoomed in" are smoothly animated over a period of 0.5 seconds, which is slow enough to avoid disorienting the user, while also fast enough to avoid slowing down the user's workflow.

2.2 Settings Objects

When the GenSession application is first launched, the scene is empty, with no nodes. The user could begin by creating an empty clip and manually entering notes in it, but a more typical usage scenario is to first create a settings object. Once created, the user can zoom in on the settings object (Figure 2) to modify its parameters.

Settings objects are used to generate clips, and contain parameters describing the kind of clips to generate. When the user is zoomed in on a settings object, the panel of options on the left side of the main window allows the user to modify the scale and number of bars in the generated clips, as well as other parameters. The example in Figure 2 shows options for generating clips with 4 bars, in E minor, using the chord progression "E- B- F♯dim E-". The chord progression defines one *target chord* for each bar, and can be entered manually, or can be inferred by the software if the user enters a cadence string like "I V II I".

Note that when generating a clip, the target chords in the chord progression are not simply copied into the clip. Instead, the target chords provide guidance for the generation algorithm, which we discuss later.

Other parameters in the settings object include: the percentage of generated notes that should fall on notes of the target chords (we call this p in a later section); the *rhythmic density*; the fraction of generated notes that should be rests (silences); and the number of notes K to generate together at each generated position in time (for example, setting $K = 3$ will cause the generated music to be a sequence of 3-note chords, that are not necessarily the same as the target chords).

Rhythmic density varies between 0 and 6, and determines the duration of generated notes (we chose the following ad hoc mapping: a density of 0 corresponds to a whole note, 1 for half note, 2 for dotted quarter note, 3 for quarter note, 4 for dotted eighth note, 5 for eighth note, and 6 for sixteenth note). In addition to setting a value for rhythmic density, the user may also set a "variance" parameter. For example, if the rhythmic density is set to 3, with a variance of 1, the resulting rhythmic density would be randomly chosen in the range 3 ± 1 for each note.

We distinguish between *global parameters*, that are constant for the entire clip, and *dynamic parameters*, that vary throughout the clip. For example, both the "percentage of notes on chords" and "rhythmic density" can be set to a single value using a slider in the settings panel on the left, in which case they behave as a global parameter. On the other hand, the user may instead draw a curve for each of these parameters with a value that varies over time, defining a dynamic parameter. For example, in Figure 2, the "percentage of notes on chords" starts off high, and then decreases to a lower value in the later bars.

2.3 Clips

Clips are the second kind of node in the scene. When the user is zoomed in on a clip, they see a piano-roll style view, within which they may manually edit notes. Figure 3 shows the rows corresponding to the target chords highlighted in grey. This highlighting can be optionally turned off. An additional option highlights all notes in the scale in a lighter shade of grey. Both kinds of highlighting can make it easier for users to position notes in the piano roll.

Every clip has 2 *voices*, or subsets of notes, that are similar to the concept of tracks. The voices are identified by color (red or blue), and the notes of a voice are displayed with this corresponding color.

2.4 Algorithms for Generating Clips

To generate new clips, the user must be "zoomed out" (i.e., not zoomed in on any node), in which case the panel of options (Figure 1) contains widgets to select and execute a generation algorithm, based on the currently selected settings object and/or currently selected parent clip.

The generation algorithms we designed were inspired in part by Povel [13]. The basic algorithm we implemented generates notes sequentially, at temporal positions denoted by $t = 1, 2, 3, \ldots$. The timing of these temporal positions depend on the rhythmic density, which may be fixed or dynamic, and which

Fig. 3. When the user zooms in on a clip, a piano-roll style view is displayed, and individual notes can be edited.

may have some "variance" allowing for random choice in note duration (i.e., distance between consecutive temporal positions). When the variance of the rythmic density allows it, the algorithm will more often choose notes of the same duration as the previous note, and will also more often choose note durations so that notes fall on the beat.

Let K be the number of notes to generate together at each generated position in time, as specified by the user in the settings object. For example, $K = 1$ means a monophonic melody is generated, and $K = 3$ means that 3-note chords are generated (not necessarily the same as the target chords). We denote the sequence of generated notes as $(n_{1,1}, \ldots, n_{1,K}), (n_{2,1}, \ldots, n_{2,K}), \ldots$, where $(n_{t,1}, \ldots, n_{t,K})$ is the set of simultaneous notes at temporal position t.

Furthermore, let p be the percentage (as specified by the user in the settings object) of notes that should fall on notes of the target chords.

The first note, $n_{1,1}$, is given a random pitch, with probability p of falling on a note of the first bar's target chord, and probability $p - 1$ of falling somewhere else on the clip's scale. From each note $n_{t,1}$, we generate $n_{t,2}$, which is used to generate $n_{t,3}$, etc., until $n_{t,K}$ is generated, at which point $n_{t,1}$ is used to generate $n_{t+1,1}$, which is used to generate $n_{t+1,2}$, etc.

Each time the first note $n_{t,1}$ at a new temporal position t is generated, a 50% coin toss decides whether the other notes $n_{t,2}, \ldots, n_{t,K}$ will have progressively increasing or decreasing pitches. Assume that increasing pitches have been chosen, for the sake of illustration. The algorithm then searches upward from $n_{t,1}$ for the *next* pitch that is either on the bar's target chord or on some other note of the scale (depending on the outcome of a p-weighted coin toss), and assigns

this pitch to $n_{t,2}$. This is repeated, searching upward from each $n_{t,i}$ to find the next pitch to assign to $n_{t,i+1}$, repeating the p-weighted coin toss each time.

Once $n_{t,K}$ has been generated, the algorithm then generates $n_{t+1,1}$ from $n_{t,1}$, the same way it generated $n_{t,2}$ from $n_{t,1}$: a 50% coin toss to choose whether to move upward or downward in pitch, and a p-weighted coin toss to determine whether $n_{t+1,1}$ will fall on the bar's target chord or on some other note of the scale. (Note that, from temporal position t to $t+1$, we may have moved into a new bar with a new target chord, or we may still be in the same bar.)

The above algorithm can be executed to create a new clip from a settings object. There are also 3 variants of the above basic algorithm:

Variant 1: "Keep existing rhythm, generate new pitches": this reuses the rhythm of an existing selected clip, and generates new pitches in the newly generated child clip. The child clip is then shown linked with arrows to both its parent settings object and parent original clip.

Variant 2: "Keep existing pitches, generate new rhythm": similar to the first variant, this results in a child clip with two parents: a settings object, and the original clip. In this case, the total duration of the modified notes may no longer match the total number of bars, so the algorithm will compensate by either truncated the end of the child's notes if they extend past the last bar, or fill in the end with generated pitches if the modified notes don't reach the end of the last bar.

Variant 3: for each note in a parent clip, this variant randomly chooses to either change the note's pitch, or the note's duration, or change both, or change neither. The probabilities for each outcome are currently fixed in the source code, but could easily be exposed with sliders similar to the other generation parameters.

Finally, with each of the 4 generation algorithms above (the basic algorithm, and the 3 variants), the user has the choice of having each voice (red or blue) in the generated child clip be copied from the parent clip or generated with the algorithm.

2.5 Timeline

In Figure 1, along the top of the 2D scene, there is a timeline widget that can be used to play a sequence of consecutive clips. The user can drag clips into the timeline in any order, and hit "Play" to listen to the resulting piece.

2.6 Additional Features

The user may select two existing clips, and combine them into a single child clip, made with the blue voice of one parent and the red voice of the other parent (Figure 4).

Each clip also has a "regeneration" option that, when turned on, causes the content of the clip to be regenerated on-the-fly when the clip is played. Such clips can be dragged into the timeline between other clips with fixed content, in

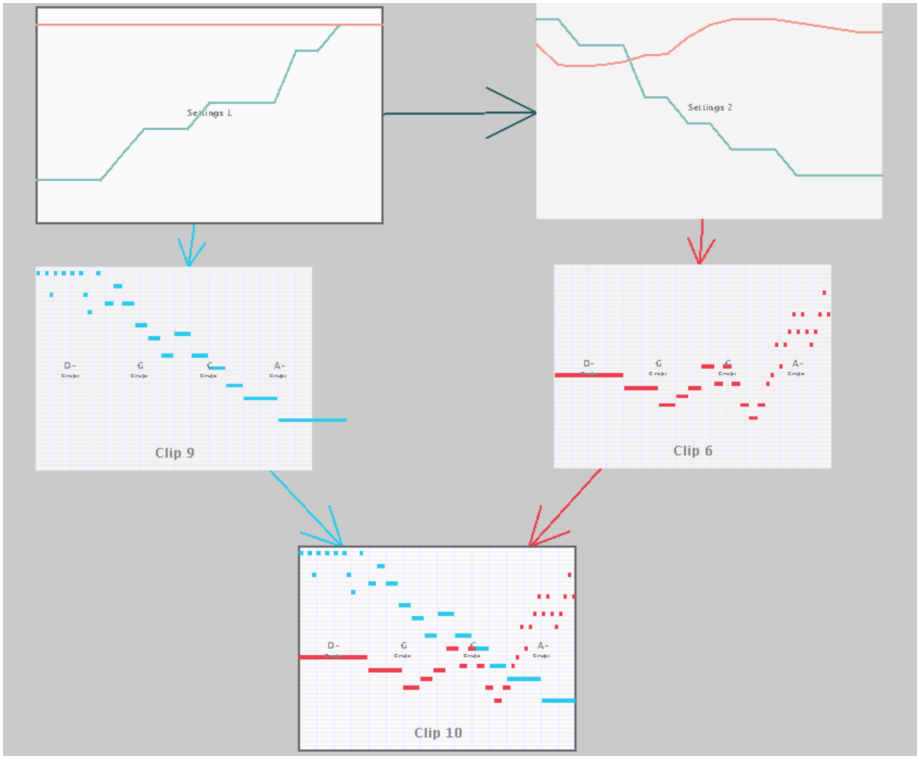

Fig. 4. The user has defined two settings objects, one with an increasing rhythmic density, the other with a decreasing rhythmic density, and generated one clip with each of them. Next, the user combines the two clips into a child clip, made with the blue voice of one parent, and the red voice of the other.

which case they behave somewhat like an improvised clip that sounds different each time the piece is played (but always with the same target chord sequence).

Once a clip has been generated, the user can experiment with changes to the scale or changes to the (target) chord progression, causing individual notes to update. This is done using the panel of widgets on the left of Figure 3, after zooming in on the clip. For example, if the scale is initially C major, and the first bar's target chord is C major, the user can change the first bar's target chord to C minor, in which case all the E notes in that bar are changed to E♭. Alternatively, the user may change the scale from C major to C (natural) minor, in which case all E notes in the bar become E♭, and all B notes in the bar become B♭.

Finally, GenSession can save MIDI files, as well as output a live MIDI signal to a MIDI port, with each voice on a different channel. This allows the use of GenSession in conjunction with other tools such as Ableton Live.

2.7 Source Code and Video Demonstration

The source code for our prototype, as well as a video demonstrating its features, can be found at
`http://hifiv.ca/~francoiscabrol/GenSession/`

3 Initial User Feedback

To evaluate our interface, one of us (Cabrol) conducted individual meetings with 7 users having some professional relationship with music: 3 master's students in music and technology, 1 master's student in acoustics, 2 Ph.D. students doing research related to music and technology, and 1 artist who has performed experimental music at concerts. 2 of these users are experienced musicians, 3 of them are less experienced intermediary musicians, and 2 of them have very limited knowledge of music theory. None of them had seen our prototype before the meetings.

We started the meetings by demonstrating the main features of the prototype for approximately 15 minutes. This demonstration was the same for each user, and involved generating clips with a melody on the blue voice, then generating clips with rhythm chords on the red voice, and then combining clips to merge together the melodies and chord accompaniment into a single clip.

Next, the participant was invited to freely interact with the prototype. In most cases, the meeting lasted a total of 1 hour, but most users were interested in using the prototype longer, or using it again at a subsequent meeting, or in obtaining a copy of the code. Two users spent a total of 2 hours each using the prototype.

When using the prototype, users started with an empty scene. They were invited to create anything they like, to explore and do as they wished, with help being provided by Cabrol whenever the user needed it. Most users roughly recreated the steps that had been shown to them in the demonstration: creating settings objects, then generating clips, then combining different voices of clips. However, each user chose different scales and chord progressions, and different users also played with the rhythmic density in different ways. A critical step for the user to generate clips is to first choose a scale and (target) chord progression in the settings object. Novice and intermediate users found this step challenging, and all users suggested having an easier way to do this, for example, an interface that would suggest chord progressions, or one that would allow the user to hear previews of chords and then drag them into a progression.

Users liked the zoomable user interface (ZUI) a lot, and they liked seeing thumbnail representations of the clips when zoomed out, enabling them to distinguish clips more easily. The automatic re-framing mode was also very well-liked. One user stated that they would very much like to have a similar feature in another music editing program that they use regularly.

All users found the prototype "easy to use", but also found that it took some time to learn the various features of the user interface. After 30-45 minutes of

use, most had learned the main features, and were interested in using it given more time and opportunity. As one user summed up, "Good interface, easy to use, but requires a training time like every music production tool to understand all the functions." Generally speaking, the users with beginner and intermediate experience in music found that the prototype allowed for easy creation of interesting sounding pieces, whereas the more experienced musicians thought the tool would be appropriate for experimental use and for teaching musical concepts to others. 4 of the 7 users stated they would like to have a similar application for themselves to compose or just to experiment.

4 Future Directions

As indicated by the user feedback we obtained, it would be useful to have a mechanism to make it easier to hear previews of chords and progressions, and possibly even automatically suggest chord progressions. Automatic suggestions might be generated in a partially stochastic fashion, possibly based on parameters describing the desired "tension".

Future work could allow the user to connect the GenSession user interface to other generation modules, possibly defined in external software, or possibly through a plugin architecture. Rule-based algorithms [8] and genetic algorithms [6, 15] would both be useful sources of material. As Papadopoulos and Wiggins [12] state, "Systems based on only one method do not seem to be very effective. We could conclude that it will become more and more common to 'blend' different methods and take advantage of the strengths of each one."

To help users manage large collections of clips, features could be added for performing automatic layout on demand, based on graph drawing algorithms [7]. Users might also be allowed to select groups of clips and collapse them into a "meta node". Additionally, techniques from virtual desktop interfaces that enable users to collect icons together into "piles" [10, 1] could be adapted for working with sets of clips. This leads to a related question of how to provide the user with a meaningful "musical thumbnail" of the contents of a pile of clips: perhaps when the user hovers their cursor over a pile or collection of clips, a randomly chosen clip, or intelligently-chosen subset, could be played.

Finally, features to help users understand the differences between two clips could be beneficial, perhaps by highlighting the different notes between a pair of chosen clips, similar to how different versions of a text file can be compared by visual "diff" tools. Highlighting differences in individual notes could help the user immediately see where the clips differ without having to listen to a playback of both clips. In addition, highlighting differences in chord progressions or keys between two clips could help the user check if two clips are "compatible" before merging them into a single clip. Difference highlighting could be done in response to cursor rollover: the clip under the curser, and all its neigboring clips, could have the differences in their notes highlighted. Differences could also be visualized at the level of entire collections of clips: users may benefit from a visualization showing how a scene has evolved over time, i.e., showing which clips have been

deleted or created. Techniques for this could be adapted from previous work on the visualization of dynamic graphs [2, 16].

Acknowledgments. We thank the researchers and musicians who gave us their time and feedback. This research was funded by NSERC.

References

1. Agarawala, A., Balakrishnan, R.: Keepin' it real: Pushing the desktop metaphor with physics, piles and the pen. In: Proceedings of ACM Conference on Human Factors in Computing Systems (CHI). pp. 1283–1292 (2006)
2. Archambault, D., Purchase, H.C., Pinaud, B.: Animation, small multiples, and the effect of mental map preservation in dynamic graphs. IEEE Transactions on Visualization and Computer Graphics (TVCG) 17(4), 539–552 (2011)
3. Ariza, C.: An Open Design for Computer-Aided Algorithmic Music Composition: athenaCL. Ph.D. thesis (2005)
4. Assayag, G., Rueda, C., Laurson, M., Agon, C., Delerue, O.: Computer-assisted composition at IRCAM: From PatchWork to OpenMusic. Computer Music Journal 23(3), 59–72 (1999)
5. Bederson, B.B., Hollan, J.D.: Pad++: A zooming graphical interface for exploring alternate interface physics. In: Proc. ACM Symposium on User Interface Software and Technology (UIST). pp. 17–26 (1994)
6. Biles, J.A.: GenJam: A genetic algorithm for generating jazz solos. In: Proceedings of the International Computer Music Conference (ICMC). pp. 131–131 (1994)
7. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice-Hall (1999)
8. Gwee, N.: Complexity and heuristics in rule-based algorithmic music composition. Ph.D. thesis, Department of Computer Science, Louisiana State University, Baton Rouge, Louisiana (2002)
9. Laurson, M., Kuuskankare, M., Norilo, V.: An overview of PWGL, a visual programming environment for music. Computer Music Journal 33(1), 19–31 (2009)
10. Mander, R., Salomon, G., Wong, Y.Y.: A 'pile' metaphor for supporting casual organization of information. In: Proceedings of ACM Conference on Human Factors in Computing Systems (CHI). pp. 627–634 (1992)
11. McCartney, J.: Rethinking the Computer Music Language: SuperCollider. Computer Music Journal 26(4) (2002)
12. Papadopoulos, G., Wiggins, G.: AI methods for algorithmic composition: A survey, a critical view and future prospects. In: AISB Symposium on Musical Creativity. pp. 110–117 (1999)
13. Povel, D.: Melody generator: A device for algorithmic music construction. Journal of Software Engineering & Applications 3, 683–695 (2010)
14. Simon, I., Morris, D., Basu, S.: MySong: automatic accompaniment generation for vocal melodies. In: Proc. ACM Conference on Human Factors in Computing Systems (CHI). pp. 725–734 (2008)
15. Unehara, M., Onisawa, T.: Interactive music composition system-composition of 16-bars musical work with a melody part and backing parts. In: IEEE International Conference on Systems, Man and Cybernetics (SMC). vol. 6, pp. 5736–5741 (2004)

16. Zaman, L., Kalra, A., Stuerzlinger, W.: The effect of animation, dual view, difference layers, and relative re-layout in hierarchical diagram differencing. In: Proceedings of Graphics Interface (GI). pp. 183–190 (2011)