A Distributed Interactive Music Application using Harmonic Constraint

Donald P. Pazel, Steven Abrams, Robert M. Fuhrer, Daniel V. Oppenheim, James Wright

{ pazel, abrams, rfuhrer, music, jwright @watson.ibm.com }

Computer Music Center
T. J. Watson Research Center, IBM
P.O. Box 218, Yorktown Heights, NY 10598
www.research.ibm.com/music

Abstract

We describe a novel and intuitive interactive application for musical performance. This application, called the *Pazellian*, provides an easy way for users to control musical performance aspects such as dynamics and articulation, instrumentation, and the actual pitches being played. Users interact with the music via simple gestures on a graphical user interface. "Smart Harmony" technology harmonically constrains the pitches played, allowing even users with no musical training to obtain musically sensible results. The application is distributed, allowing multiple users to perform or control different instruments together. This paper discusses the *Pazellian's* functionality and design. We further describe novel features such as aural highlighting and soloing, along with issues and solutions concerning its distributed architecture.

INTRODUCTION

Music enthusiasts who lack musical training face a dilemma: How can they have fun interacting with music without investing years in musical training? Can they be empowered to feel that they are creating music using software containing the basic musical competence needed to realize their musical ideas, without undue restrictions? Can a shared social experience be created for them? Can such a system prove interesting to experienced musicians as well? With these questions in mind, we set out to develop an interactive application for musical performance. This application has become known as the *Pazellian*.

In brief, the *Pazellian* is a distributed, interactive music application. It provides a way for users to easily control many aspects of a musical performance. For example, users can vary dynamics and articulation, the instrumentation, and the actual pitches being played. Through simple gestures on a graphical user interface, even naïve users can quickly get a sense that they are "playing" instruments using only a mouse. The application is distributed, consisting of specialized server and client software components, allowing multiple users to "play" different instruments together.

The design of such an application poses many difficult problems. Musical considerations include how to facilitate composing and performing in a way that makes musical sense. User-interface issues include ensuring that such a program speaks in a conceptually simple but effective manner, allowing naïve users to intuitively control the music. Architectural issues include how to enable and coordinate concurrent real-time musical interactions over a computer network (overcoming latency issues). The *Pazellian* addresses these considerations, and the remainder of this paper discusses the solutions we found. We conclude with a summary and a review of a small usability study.

OVERVIEW

The *Pazellian* is a distributed music application consisting of client and server components. Users at client stations interact with the server through the client's graphical interface. Clients connect to the server and either join an ongoing performance, or choose a MIDI music selection from a given set and request the server to start a new performance. Clients then use the graphical interface to control aspects of the music, either for a particular musical voice or for the piece as a whole.

A *Pazellian* user can assume one of three roles in controlling the music: Performer, Conductor, or Maestro. As Performer, the user has control over the pitch range, volume and instrument of a specific musical voice. The Conductor role permits the selection of the musical piece, the tempo, the volume, and harmonic aspects of the music. As Maestro, the user gets simultaneous control over all of the individual Performer parameters of all voices in the piece of music. These three roles offer considerable control over the music, and a surprisingly satisfying experience to the user.

The application supports two configurations with different purposes. In the localized configuration, client stations are located together near the server, which renders the MIDI music for all to hear. In the distributed

configuration, client stations are geographically dispersed; each renders the MIDI music for its user, in close synchronization with the server.

The localized configuration is ideal for group engagements, perhaps with an audience. The distributed configuration, on the other hand, is ideal for Internet-like engagements in which users participate, each in isolation. In both cases, a significant feature of the interaction is that each user hears the combined effects of all users' musical manipulations. Figure 1 illustrates the remote configuration, with the bottom three computers playing music synchronously with the server.

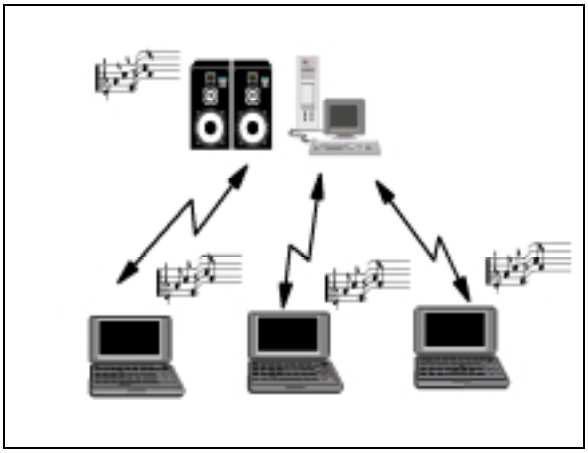

Figure 1- Pazellian Configuration

The *Pazellian* is based on Music Sketcher technology (Abrams, et al.: 1999), which derives from prior work in intelligent music composition systems (Oppenheim: 1996), (Wright: 1997). Thus, structural aspects of the MIDI music used in this technology factored into the design and enabled its capabilities. The musical composition is structured as a set of "scoreparts", each representing a performance voice on a given MIDI track, which can be dynamically assigned a MIDI instrument. The *Pazellian* utilizes our Smart Harmony system (Abrams et al.: 2000), in which individual notes in the composition are annotated with harmonic information. This information determines a set of harmonic constraints, so that pitch-shifting operations give harmonically sensible results.

As described above, the client controls the music by assuming one of three roles, each identified with a specific view in the client's interface. To change to a desired role, the user clicks on one of the icon buttons that appear in the upper left-hand corner of all three views. For example, in the performance view shown in Figure 2, the first two icon buttons select the conductor or maestro views, respectively. The additional icon buttons in this view represent the voices of the current musical composition (in this case, 5 of them), allowing the client to request control over the corresponding voice. The small icon under the Piano voice in Figure 2 indicates that a client currently has control of that voice.

PERFORMER VIEW

The concept behind the performer view (Figure 2) is to allow the client to modify the current music by taking control of an individual voice.

Figure 2 - Performer View

The performer view provides control in two major ways, namely pitch/volume modification and instrument assignment. The pitch/volume control is the large square on the left-hand side of Figure 2. The vertical axis of the square defines a two-octave pitch range, and the horizontal axis defines a volume range. By grabbing an indicator inside the square, the user changes volume and pitch simultaneously. Smart Harmony rectifies pitch changes to conform to the music's harmony. Since pitch correction is minimized, the deviation in the pitch gradient from the user's specified gesture is imperceptible.

Significantly, the *Pazellian*'s pitch control preserves the original melodic contour of a given part when the user is not actively changing the amount of pitch shifting. This maintains the fine melodic detail of the part, while still allowing the user to substantially control its overall shape and register. While actively changing the pitch shift amount, the user can largely control the shape of the melodic line, subject to the harmonic constraints noted above.

The user selects the desired instrument using the control on the right-hand side of the Performer View. The 128 MIDI instruments are represented by 8 instruments in each of 16 orchestral sections. Clicking on a square assigns the given instrument to the voice. This organization and visual cueing makes it easy for the user to quickly identify the desired instrument.

The performer view provides a creative platform through two novel musical controls: *aural highlighting* and *soloing*. These features provide unique means for bringing improvisation and individuality into the music making experience.

Aural highlighting is a technique whereby the user manipulates the pitch/volume control with that voice emphasized over the others. This gives the effect that the

client is leading and improvising. The effect is achieved by substantially lowering the volumes of the other voices while pitch/volume changes are being made on this voice. Currently, only one voice can be aurally highlighted at a time; the server highlights voices on a "first-come first-served" basis.

Soloing is a variation on aural highlighting that allows controlled entry-exit as a performer. The voice is normally muted, but while the user manipulates the pitch/volume control, the voice is un-muted and highlighted. This provides a sense of soloing with an instrument, including client-timed entrances and exits. A voice may be composed with this use in mind by creating arrangements of riff patterns that are tailored more to improvisational styles. The performer may then craft the soloing to the current character of the music.

Finally, an indicator on the bottom left of the view lights while that voice is rendering music.

THE CONDUCTOR VIEW

The conductor view (Figure 3) provides the user with control over global aspects of the music.

Figure 3 - Conductor View

The main control aspects include overall volume, tempo, and music selection. More significantly, user control is provided over the harmony of the current music. In the conductor role, the user can select a chord or key and apply it immediately to the music. This ability to assign harmony is facilitated by the fact that harmonic annotations on the musical notes are independent of chord types: the annotations identify whether the notes are chord members, assigned partials, etc. Thus, chords that are radically different from the original can be applied. The chord and key arrangement at the center of the conductor view allows the selection of a new harmony. Additionally, the user may revert to the original harmony. Smart Harmony may be disabled or enabled with a checkbox, providing aural illustration of its positive effect.

The matrix of chords and keys in Figure 3 is an early attempt to make this information accessible to users. Users found this interface to be complex, however, and it is not described here; more intuitive approaches to chord selection are being considered.

Nonetheless, the ability to dynamically assign different harmony to rendered music is a significant feature. More than simply setting the tonality of the music, dynamic harmonic manipulation was found to be an effective means for the client to experiment with and evolve harmonic textures. Especially when used with music with repetitive themes, such as with Bach, users can produce an almost hypnotic quality as they continually insert successive harmonies at the beginning of thematic beats.

THE MAESTRO VIEW

The Maestro view, shown in Figure 4, allows full control over all of the playing voices. This view provides for each voice a set of buttons, sliders, and switches for controlling various aspects of the performance of that voice. These aspects include volume, pitch, mute/unmute, and instrument. Indicators in the left-most column light up when music for that voice is actually being performed.

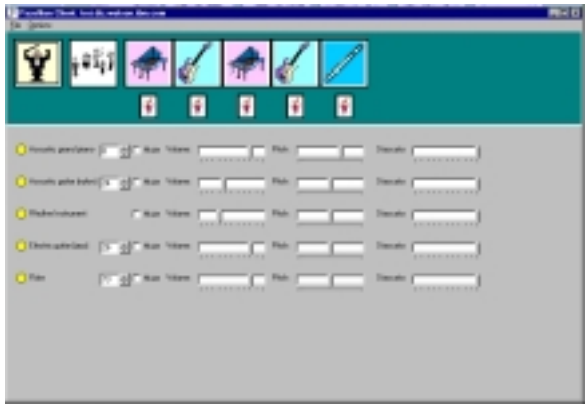

Figure 4 - The Maestro View

This view also provides sliders for manipulating the articulation (staccato/legato) of each voice. This manipulation is achieved though the use of a Music Sketcher duration modifier, in which a simple [0–1] range scales the original duration of each note.

ISSUES ON DISTRIBUTION

The *Pazellian* is a client server application. A TCP-based transport framework provides the messaging infrastructure between client and server, along with a well-defined set of messages[1]. These messages are typically small, each consisting of an appropriate set of numerical parameters for each message type. Most messages fall into two categories. Some deal with musical parameter changes, such as set tempo, volume, and pitch. Others deal with client or voice management, such as reporting when clients takes control of a voice, or updates of connection state information to clients.

[1] Due to having local high-speed network access and to having only modestly frequent client-server interactions, there was no need to consider using UDP protocols.

The server maintains global state information for the application, such as a list of clients, their roles, and which client controls which voice, and propagates that state information to the clients. For example, when a client assumes a different role, the server updates the appropriate state information internally and broadcasts it to all the clients to reflect into their views. The server also arbitrates access to various system features. For example, if two clients attempt assume a performer role on the same voice, the server determines which client receives control based on the first message it received, and disallows the second attempt.

The synchronization of music among clients in the remote version of this application was the most interesting issue that arose in making the *Pazellian* into a distributed application. This issue arose in a variety of situations, but especially when new clients attached to the server in an ongoing performance. Each client renders the music, and the server acts as a guide to the synchronization. The technique involves each client sending a message to the server requesting the server's current point in rendering. On receipt of that information, the client takes the difference between its point and the server's and adds an adjustment on estimated network latency (e.g. ½ of the round trip time for the message). The client then adjusts its rendering tempo based on the adjusted difference to gradually catch up with the server. A simple computation estimates the amount of time for the client to reach synchronization, and a timer is set for this interval. When the timer expires, the synchronization process is repeated.

This technique works quite well and over a couple of iterations, clients and server synchronize nicely. Applying this synchronization algorithm continuously is a low cost and effective way to maintain synchronization despite factors external to the application, e.g. rendering drift, or other operating system related factors that impact rendering timings. In addition, as further work, techniques involving graduated tempo changes may provide more audibly subtle means to achieve synchronization.

SUMMARY

Although incomplete, the *Pazellian* application presently demonstrates useful and novel computerized techniques for supporting musical creativity. The effectiveness of this work was explored in a brief usability study at a major theme park. Random people from diverse age groups were chosen to work with it for roughly fifteen minutes, and provided feedback on their experiences.

The overall impression of the concepts behind the *Pazellian* was quite positive. Most people were comfortable with the idea of multiple clients interacting to make music. Most found the experience enjoyable, to the point where many wanted to have their own copy of a collective performance. The effects of Smart Harmony were transparent to the user, except when demonstrated by disabling it. Users generally felt the music they were making made sense. Feedback regarding the user interface, while generally favorable, indicated that some aspects were at times confusing. For example, many people fastidiously followed the axis lines on the pitch/volume control, and had to be told to freely move the mouse over the control's area. Most people gravitated to the performer view, but a modest subset enjoyed the maestro view and the level of control it gave over all the voices. Although less emphasized during the test, the conductor view was more difficult to appreciate. The purpose and semantics of the harmony controls were unclear to all but the most experienced musicians. However, the bulk of our work was more on creative interaction with music, than on building an ultimately truly effective user interface, and so this was not such a great surprise.

There is still much more work to be done on the *Pazellian*. To begin with, further investigation is needed to find an appropriate technique for presenting users with the harmonic manipulations. In addition, we would like to add some mechanism for directly inputting music as opposed to only modifying the given music content. Of course, continued enhancement of the user interface is always a need, and there are many minor features to be added. Overall, however, the *Pazellian* is a useful step forward in understanding how to bring an interesting and enjoyable musical experience to a wide audience.

ACKNOWLEDGEMENTS

We would like to thank Lauretta Jones for her help and support during this project, Donna David for her artwork contributions, and Rick Hughes for his assistance arranging the usability studies.

REFERENCES

Abrams, S., Oppenheim, D., Pazel, D., Wright, J. 1999. "Higher-level Composition Control in Music Sketcher: Modifiers and Smart Harmony." *Proceedings of the ICMC*, Beijing.

Abrams, S., Fuhrer, R., Oppenheim, D., Pazel, D., Wright, J. 2000. "A Framework for Representing and Manipulating Tonal Music." *Proceedings of the ICMC*, Berlin.

Oppenheim, D. 1996. "DMIX—A Multi Faceted Environment for Composing and Performing Computer." Computers and Mathematics with Applications 32(1): 117–135.

Wright, J., Jameson, D., Oppenheim, D., Pazel, D., Fuhrer, R. 1997. "CyberBand: A 'Hands-On' Music Composition Program." *Proceedings of the ICMC*, Thessaloniki.

