Partial Reflections: Interactive environments for musical exploration

Andrew Johnston, Ben Marks, Linda Candy & Ernest Edmonds

This paper describes an ongoing project to develop interactive environments for musicians that encourage musical exploration. A process of developing software such as this, where requirements are highly dynamic and unclear is outlined and two musical compositions and associated interactive environments entitled 'Partial Reflections' are described.

In this paper we describe ongoing work to develop interactive environments, targeted at musicians, that encourage musical exploration. This work is practice-led, in that we aim to identify successful and unsuccessful design criteria by developing these environments concurrently with musical compositions, reflecting on the process of their design and having musicians provide feedback on their effectiveness. This paper describes some of the strategies we have used to attempt to engage users of our software and stimulate creative exploration.

1. Past work

There are, of course, a very large number of applications designed for musicians, including software for editing recorded music, transforming live music in real-time, notating music, teaching aural skills, etc, etc. Many of these applications are in wide use in many contexts, including recording and teaching studios, classrooms, live performances and individual practice. Therefore, in order to clarify the focus of the work described in this paper, we state here that we are interested in software that:

- responds in real-time. That is, it responds directly and immediately to live audio input;
- provides some kind of audio-visual representation of or response to the music;
- is suitable for use by individual musicians in performance and practice;
- encourages musical exploration; and
- may also be suitable for use as a teaching aid.

There is a body of past work that broadly fits these categories although

it is not as comprehensive as the body of work in areas such as digital recording and editing software for example. It is possible to group existing real-time musical/visual software into two broad categories. Firstly, there are those which attempt to provide unambiguous feedback to the musician so that they may discover characteristics of their playing they were unaware of. This software acts like a teacher who "tells it like it is" as the musician plays. Just as an electronic tuner, for example, provides musicians with feedback on whether or not they are in tune relative to the equal-tempered scale, computer software can provide this information and more on screen. Such software usually produces visual representations of the musical input in the form of graphs, spectrograms and other well-known mathematical visualisations.

The program "Sing and See" (Thorpe 2002) uses this style of display, including real-time graphs showing pitch and harmonic characteristics of live audio. As the name suggests, this software is intended to provide singers and their teachers with visual feedback to enhance their understanding of their performances. Other programs such as Voce Vista (Miller and Schutte 2002) and WinSingad (Langner et al. 2000) have similar features and visual style.

The second type of software has less defined goals. Whereas the 'unambiguous feedback' software aims to provide the musician with impartial measurements of their playing, this software is instead intended to stimulate musical exploration and responds to live music in more unusual, evocative, and ambiguous ways. Software of this type is sometimes found in interactive artworks or electro-acoustic performance pieces.

An example is the "Singing Tree" by Oliver et al. (1997), which, like "Sing and See", provides a visual response to live singing but with a very different style. Singers using this program (designed as an interactive installation for a gallery) sing into a microphone which is connected to a computer. If the singer produces a stable pitch, then a video is played which gives the impression of moving forwards towards an identifiable goal- an opening flower for example. If a steady pitch is not maintained the video reverses. In addition to the visuals, audio accompaniment is also provided- the steady pitch being accompanied by consonant string, woodwind and vocal harmonies and unstable pitches by dissonant, percussive sounds. This style of software is specifically intended to encourage a playful approach to music making.

We do not suggest that one style is superior to the other and expect that many musicians will find a place for both styles of software in their practice. No two musicians approach their art (and craft) in exactly the same way, and individual musicians take different approaches at different times in their practice, so there is unlikely to be one 'super tool' which will suit all musicians at all times. Our work, however, is of the

type that aims to stimulate exploration rather than provide unambiguous feedback.

2. Goals of our work.

Our aim is to develop software that provides a supportive environment for musical exploration. We use the broad term 'musical exploration' deliberately so as to include improvisation as well as more formally composed music. Also, this exploration may take place in different contexts - in the privacy of the practice studio, in live performance or in teaching for example.

The work we describe here is a result of a collaboration between a composer/musician (the 'composer') and a technologist/musician (the 'technologist'). The composer has a masters degree in composition and performance, specialising on the trombone. In addition he has extensive professional experience in a wide range of musical ensembles, including contemporary music ensembles and symphony orchestras. The technologist has an undergraduate degree in music performance (also trombone) and professional performance experience in many ensembles. He has also completed a masters degree in computing and currently works as a lecturer in a faculty of Information Technology.

An important realisation at the outset was that there are no clear pre-existing requirements for software of this kind. Literature on creativity support provides high-level guidelines (eg. Nickerson 1999) but this is not a well-defined problem awaiting technical solution. Rather, the problem of developing software for encouraging exploration in musicians would itself seem to demand an exploratory approach.

Because of this, we take a design approach that draws on participatory design and agile software development techniques and philosophies. Namely, we engineer a situation where musical compositions and software co-evolve. This works as follows:

1. Potentially interesting ideas are brainstormed.
2. Initial simple prototype 'sketches' of interactive musical environments are produced.
3. The composer explores these sketchy prototypes by improvising with them and developing the improvisations into initial compositional sketches.
4. Further refinements to the software are suggested by the composer as he develops his composition and as these refinements are implemented, they in turn cause him to modify and expand on the composition.

The outcome of this iterative process is a composed piece of music for trombone and interactive software. In addition, by carefully

documenting and reflecting on the process we identify characteristics of the design of the software that seem to aid musical exploration and engagement.

This exploratory design method is in contrast to the 'waterfall' approach to software development where the requirements and design are decided up front and the software development does not commence until it has been decided exactly what will be built. This approach is problematic in situations such as ours when requirements are highly dynamic and unclear, as evaluation and feedback can only take place once the software has been completed. The iterative approach to development instead recommends developing the software in small chunks and refining the design and requirements based on frequent feedback and direct design input from users. This is the approach recommended by the so-called 'agile' software development methods (Beck 2000, Cockburn 2002) among others.

3. Use of virtual physical models

We began to use physical models as the basis of our software very early on in the project. The term 'physical models' in this sense does not refer to actual physical objects that exist in the real-world, but rather to the building of 'virtual models' made up of masses and links that move around the computer display.

During the development of our work work, we hit upon a particular style of interactive music software which made use of physical models (Momeni 2006, Henry 2004). In this approach, the software incorporates a physical model which may be thought of as a kind of *virtual sculpture*. The software designers build the sculpture by positioning various masses in virtual space and specifying their physical qualities such as mass. They may also link these masses together with virtual 'springs' of certain lengths, rigidity, etc. Because this sculpture obeys the laws of Newtonian physics, it responds in ways that appear natural when forces are applied to it. In our case, the forces are mapped to characteristics of the music that is played. So, for example, if the loudness of the input sound is mapped to the quantity of force exerted on the sculpture then playing a loud note will cause a large amount of force to be applied to the model and, depending on it's structure, it may bounce around the screen, change shape and so on. In our work, these movements also cause the computer to output sounds.

To put it simply, the musician's live sound exerts force on parts of the physical model/virtual sculpture which causes it to move in physically plausible ways. Figure 1 shows a high-level view of how this works. Note that while it does not necessarily have to be the case, in our work the visual output is a direct representation of the physical model itself.

The intention is that the musician has a feeling of direct control over the `virtual sculpture' with their playing and that the audience (in situations where there is an audience) can readily perceive this.

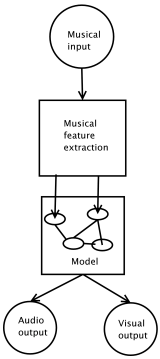

Figure 1 – Block diagram showing the use of a physical model to map between musical input and audio/visual output.

We took the decision to use physical models as the basis of our software for a number of reasons. The most important in terms of providing an engaging experience for musicians using the software appear to be:

- The musician has a feeling of control over the visuals, and through the visuals, the sounds produced by the computer;
- There is a readily apparent link between the acoustic sound and the computer-based audio-visuals[1];
- Because the models respond in the same way as everyday physical objects, the behaviour of the models is intuitively understandable by the musician;
- Because the movements of the model which produce sound are based on realistic physical motions, the resulting sounds have an 'organic' quality.

4. Partial Reflections I

The piece we describe in this paper is a two movement work entitled 'Partial Reflections'. The first movement of this work has been described in detail elsewhere (Johnston, et al 2005), but we provide a brief overview here to contextualise the fuller description of the second movement which follows. The virtual physical model which forms the basis of the software for this movement is a simple structure comprising 12 spheres linked together by elastic 'springs' (figure 2).

Figure 2 - The physical model for Partial Reflections I

Each sphere in the model is associated with a particular pitch-class (or note) and by playing into the microphone connected to the computer the musician can exert forces on the model. If the player plays a G for example, then the sphere at the bottom of the model is pushed with a force that is proportional to the volume of that note. When no notes sound the model returns to its resting position, hanging down from the top of the screen.

Movements of the spheres also cause audio to be output by the computer. The software stores the frequency of each note played by the musician and this frequency is associated with the appropriate sphere. For example, if the musician plays an A with a frequency of 440Hz, then the A sphere is assigned that frequency. If they subsequently play a lower A with a frequency of 220Hz then this replaces the value of 440Hz previously assigned to the A sphere. As the spheres move around, they generate pitches at their assigned frequency. The faster they move, the louder their pitch sounds.

The effect of this is that by carefully selecting pitches and volumes the composer can influence the timbre (or tone) of the computer output. As such, the music which was composed for this software and trombone (or, perhaps more precisely, trombone augmented by this software) made much use of mutes and instrumental techniques such as multiphonics in order to generate many different timbres and explore the effect of these on the visual behaviour of the model and on the resulting sounds.

5. Partial Reflections II

Overall, the first movement is slow and smooth and emphasises changing timbres. In contrast, we wanted the second movement to have strong rhythmic drive, be energetic and generally have a 'spikey' character. The decision was therefore made that, unlike the first

movement with its emphasis on longer notes with changing timbres, the software for second movement would ignore all sounds other than the very beginnings of notes - the moment of articulation or 'attack'. Because the trombone has a very large range of potential styles of articulation, this still leaves the musician with considerable room for exploration.

The underlying physical model for the software for this movement is again quite simple, and again comprised 12 spheres - each one associated with a pitch-class[2]. Each sphere is linked to a fixed central point with a very short 'spring'. Figure 3 shows the arrangement. The model is configured so that as soon as the software is started the 12 spheres begin oscillating very rapidly about the fixed point. In this initial state the length of the springs is so short that it is not really possible to see how many spheres there are (see figure 4).

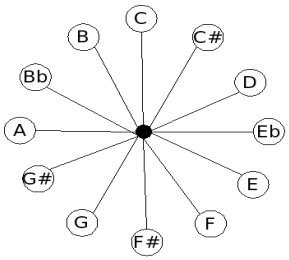

Figure 3 - Physical model for Partial Reflections II.

Figure 4 - Photograph of performer and visual display just before playing commences.

When the performer plays, the computer detects the onset of the note (ie. the moment of articulation) and determines its pitch class and volume. These are used to direct force onto the physical model, with the pitch-class selecting which sphere will have force applied to it and

the volume determining how much force is applied. The louder the detected note, the greater the force.

Force is always directed in the direction that will cause the sphere to spin anticlockwise. For example, if the sphere is currently positioned at "3 o'clock" then the force will be directed in an upwards direction. If it is positioned at "2 o'clock" then force will be directed both an upwards and to the left. The effect is that by playing notes the musician can spin the spheres anticlockwise around the fixed point. The louder they play each note, the faster the associated sphere will spin.

As a result of the spheres spinning more rapidly, they are pushed out from the central point. To illustrate, if the performer were to play a series of short, loud Cs, then the C sphere would rapidly accelerate and move out into a higher orbit around the central point. When the performer stopped playing Cs, it would gradually spin back down to the central point. (In our model the friction is very low, so the spheres rarely if ever stop spinning completely. When force is stopped they instead obit rapidly at a very low altitude around the central point.)

As with the first movement, the software for the second movement produces sounds. We have stated earlier that the pitch-class of each note is identified and that each spinning sphere is associated with a particular pitch-class. The first 100ms of each note is recorded by the software and is linked to a sphere. Each time the sphere completes a half turn, the software plays back the recorded sound linked to that sphere with one additional modification: the higher the orbit, the slower the playback. The effect of slowing the playback is to lower the pitch of the played-back note. So, if the sphere has a very high orbit (because it has had a lot of force exerted on it) then the note that plays back every half rotation will be pitched quite low.

An example may help to clarify this behaviour. When the software starts the spheres are spinning rapidly around a central point at a very low altitude. If the performer plays a Bb several things happen:

1. The Bb sphere has force exerted on it in proportion to the volume of the Bb;
2. The first 100ms of the attack are recorded and associated with the Bb sphere;
3. In response to the force, the Bb sphere is pushed out into a higher orbit;
4. Every half turn, the 100ms of recorded Bb is played back, but with pitch shifted down by an amount proportional to the distance of the sphere from the central point;
5. When the performer stops playing Bbs the Bb sphere will gradually spin back into the central point and as it does so the pitch will gradually increase.

Musically, the notated composition developed along with this software is fast-paced and makes use of many different types of articulation. In addition to pitched notes, the performer also makes use of unpitched percussive attacks at times (such as in the top line of music in figure 5).

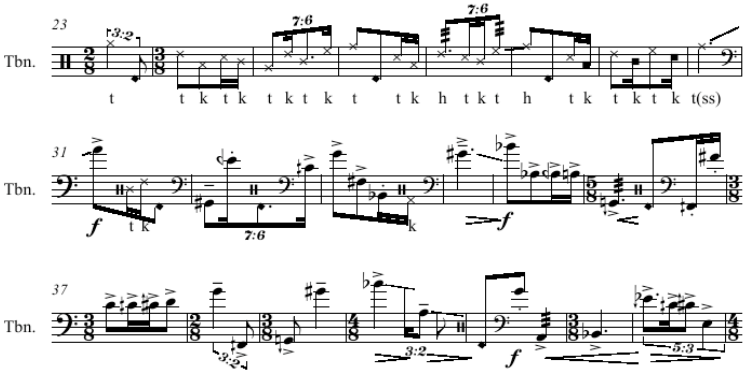

Figure 5 - Excerpt from the second movement of 'Partial Reflections' for trombone and interactive virtual sculpture..

6. Conclusions and future work

We have described in some detail two musical works that we have developed as part of a project to design interactive environments that encourage musical exploration. In the process, we have discussed the need to take an exploratory approach to the design of the software itself and have outlined some of the design criteria that have emerged so far.

Future work will involve evaluation of the existing software by a number of musicians to see whether the environments we have developed really do encourage musical exploration.

Notes

1 The compositions that emerged from the collaboration could from this perspective be seen as solo works for 'augmented trombone' - a trombone that has direct and clearly discernible sonic links to objects on screen, which in turn have links to sounds produced electronically.
2 It is of course quite possible to divide the octave into more or less than 12 pitch classes. To date we have stuck with tradition, but this is a likely area for future experimentation.

Acknowledgements

Our thanks to the developers of Pure Data, GEM and Physical Modelling for Pure Data (PMPD) for creating the software which made this work possible.

This research/work was partly conducted within the Australasian CRC for Interaction Design, which is established and supported under the Australian Government's Cooperative Research Centres Programme.

References

Beck, K. (2000), Extreme Programming Explained: Embrace Change, Addison-Wesley, Reading, MA.

Cockburn, A. (2002), Agile Software Development, Addison-Wesley.

Henry, C. (2004), Physical Modeling for Pure Data (PMPD) and Real Time Interaction with an Audio Synthesis, in Sound and Music Computing '04.

Johnston, A., Marks, B. & Edmonds, E. (2005),`Spheres of Influence' - An Interactive Musical Work, in Yusuf Pisan, ed., Interactive Entertainment (IE2005), Creativity and Cognition Studios Press, Sydney, Australia, pp. 97-103.

Langner, J., Kopiez, R. & Stoffel, C. (2000), Realtime analysis of dynamic shaping, in 6th International Conference on Music Perception and Cognition.

Miller, D.G. & Schutte, H.K. (2002), Documentation of the Elite Singing Voice, online at http://www.vocevista.com/contents.html

Momeni, A. & Henry, C. (2006), Dynamic Independent Mapping Layers for Concurrent Control of Audio and Video Synthesis, Computer Music Journal 30(1), 49-66.

Nickerson, R.S. (1999), Enhancing Creativity, in Robert J. Sternberg, ed., Handbook of Creativity, Cambridge University Press, Cambridge, pp. 392-430.

Oliver, W., Yu, J. & Metois, E. (1997), The Singing Tree: Design of An Interactive Musical Interface, in Proceedings of Designing Interactive Systems, ACM Press New York, NY, USA, Amsterdam, The Netherlands, pp. 261-264.

Thorpe, W. (2002), Visual feedback of acoustic voice features in singing training, in Proceedings of the 9th Australian Speech Science & Technology Conference, 3-5 December 2002, Melbourne, pp.349-354.