Bournemouth University
Faculty of Science and Technology
3th April 2017

Chris Lynden
BSc Music and Audio Technology

Creative Music Technology: Interactive Music Systems

Introduction

The system evaluated in this report is an algorithmic composition system created in Max MSP, influenced by user musical and computer mouse input, and driven by an orbital motion simulation written in Processing.js.

System Overview

A 16-bar sequence of MIDI data is recorded into into Max, which is split into 31 segments of varying size.

The Processing.js system simulates the orbits and motion (based on Kepler's first law) of 2 entities moving amongst 31 'planets'. Processing sends data from the simulation to Max over OSC. Each of the planets represents one of the 31 segments of MIDI data.

The proximity of the white entity to each of the 31 planets modifies the relative velocity of the MIDI segment associated with that planet. Each planet has a surrounding 'field' in which the velocity will increase from zero to the recorded values as the entity approaches the planet.

The red entity roams space, and upon collision with a planet, causes one of many modulating functions to be applied to the respective MIDI segment. (For details on currently included functions see Appendix.)

Each function has an associated weighting which modifies the likeliness of its' execution. This data is changeable during run time and is found in 'modulateSegments.js'. The code has been developed to make the addition or modification of functions straightforward and possible during run-time.

The user can influence the motion and position of the entities within processing. The left mouse button causes a strong gravitational force to occur at the cursor. With shift held, the left mouse button causes the white entity to jump to the cursor position and become drag-able. Clicking and dragging with the right mouse button allows modification of the radius of the planets' surrounding 'fields'. The space bar causes both entities to lose all velocity, holding their position.

Intention

The first consideration in designing this system was of what makes a collection of sounds or notes identify as musical content and work as a composition. Pattern recognition is a fundamental part. Music is a medium of communication, and like spoken language, code and visual communication, has to operate within expected syntax and structure while conveying new ideas intended from the composer. Any communication that is spontaneous, random or does not adhere to expected structure, phrasing and context, will struggle to convey useful information to the receiver.
However, composition that is devoid of deviation from what is expected will undoubtedly lack evidence of any creative innovation and simply appear to imitate the previous works and (musical) paradigms that the expectations are derived from.

How does one define whether creative content is representing new ideas or simply imitating, modulating and re-implementing existing ones?
Boden (1998) defined three categories of creativity, which can be applied to evaluate the artistic or intellectual processes and output of automated systems and humans alike:

- Combinatorial – producing novel combinations of familiar ideas.
- Exploratory – generating new ideas from the exploration of structured conceptual spaces.
- Transformational – involves transforming one or more of the dimensions of the conceptual space, so that new ideas can be generated that could not have arisen before.

Boden states that exploratory creativity is most commonly achieved in computer systems: while musical conceptual spaces are tricky to define computationally, combining such ideas is more challenging since the intention is to imitate a human's ability to associate memories. Once a conceptual space is defined in a system, it is straightforward to procedurally explore it.
Transforming and re-defining the dimensions of a space however, transcends what the intended functionality of any computer system is, and poses great programming challenges.

Some composers compose without instruments, perhaps beginning with just one notated idea and compose an entire symphony without hearing its progress. While some have perfect pitch and are able to audiate pitch and timbre accurately, often, numerical systems and transformations are employed where the composer chooses, due to his/her experiences. Many compositional devices can be employed without the need for instruments and improvisation, but through mathematics and patterns. Winkler (1998) states that these processes can be implemented through software algorithms and interactivity provides a platform for their exploration.

Generative systems may be able to produce surprising and coherent results when given sensible constraints, but to produce material that has inherent phrasing and structure is difficult due to the task of defining phrasing and structure computationally. Phrases within (traditionally) successful compositions tend to compliment and somewhat imitate each other, while each presenting a musical idea of their own. Copying and simply repeating previous parts will provide a clear structure through repetition but is not typical of novel or interesting music.

One concept explored prior to this development was the behavior of systems given freedom to explore a simple set of rules. Simulations of physics and swarm theory in two-dimensional space showed that novel structure might arise where it is unexpected. One sketch with Processing.js featured ellipses within a square, each attracted through space to one another. Over time they would form multiple interacting structures, beyond prediction, that would evolve throughout run-time.

In composition, a set of rules can aid productivity, but at the cost of novelty in the product. If a group of musicians are given musical constraints (I.e. tempo, key, chord selections, genre) for a composition, the pieces they produce will adhere to the intentions of the rule-maker and likely compliment each other, and require far less time and deliberation to conceive. Their creativity has been restricted however; if the rules can be considered as boundaries of the conceptual space, any transformational creativity would break the set rules by definition and the more rules that are applied reduce the scope for exploratory creativity. The same can be considered in interactive computer composition - fewer rules: greater freedom to explore: more novelty and creativity present in the result. However, a human composer will usually apply their own rules to a compositional process when not provided with any, giving the piece coherence and structure. A computer cannot do this so easily, and not without instruction.

A summary of the intended outcomes of this development

- Utilise user input to provide musical ideas, and the constraints for, automated composition.
- Execute compositional transformations on musical data.
- Produce coherent and structured musical output, reflecting the ideas input by the user.
- Utilise a system that produces incalculable yet structured results from a simple rule set.
- Produce musical output that reflects a simulation of creativity.
- Provide ways to interact with and influence the composition during run-time.

Evaluation

The system produces coherent musical output, and evolves ideas indefinitely, often producing surprising and unexpected results. The result reflects some musical intention of the user, while remaining within constraints formed from the user input and built into the system. The Processing.js sketch provides a way for the user to interact with the orbital motion system, which affects the transformation and playback of musical ideas. The code architecture enables run-time experimentation with modulating functions.

The input performance greatly affects the resulting output. Recordings that feature strict tonality and wide ranges of pitch, interval, rhythm, harmony and dynamics generally produce the most entertaining and interesting results. However, the constant modulations mean simple input can become complex output over time; varied and skilled input can become simple output.

Short phrases are reflected in the output, although any structure beyond this is not. The motion simulation provides a changing higher-level structure over time. User interaction with it gives some control over this. However, the resulting high-level structure rarely reflects any intention of such in the inputted data.

The output features the reorganization and modulation of inputted MIDI data in a musical manner; this would satisfy Boden's (1998) first classification of creativity.

As the program runs, segments of data are continually and randomly modulated; after 10 minutes of run-time the result bares little resemblance to the input. The micro phrasing within the piece is generally preserved, and the system may not produce scale degrees that were not input. Some modulating functions have constraints: when they try to make a segment's pitch too high, they will lower it instead; when they try to make notes' durations too small, they will increase them. While the system is not actively exploring musical ideas - not making informed choices, but executing functions due to events and probabilities - it will produce material that could not have been predicted yet is trapped within the constraints, or conceptual space, provided.

Rowe (1992) states interactive software models human understanding and response to simulate intelligent behavior. The segmentation and modulation of MIDI data is designed to do so in a similar way to how a composer might develop a musical idea, although the system does not simulate intelligence or analyse input to be able to truly identify the users' intentions.

Rowe (1992) also outlines a classification system for interactive systems, in which this system classifies as employing sequenced and transformative methods in its response to input.

Considering Winkler's (1998) definitions of the components of an interactive system, the produced system satisfies each, but with varying success:

- Human input is achieved through MIDI input and mouse interactions. More control during run-time would increase interactivity. An improvement would be the ability to add new MIDI data into the system during run-time. This was intended but presented programming difficulties.
- Computer listening, performance analysis and interpretation is somewhat implemented. The system defines available pitches from the performance and uses this data for modulation. The rhythm, intervals and phrasing of notes will often be preserved and reflected in later modulations. A large amount of musical data can be generated from the input. More control over the modulation of segments through input analysis would benefit the system.
- The computer composition elements can generate interesting and surprising results from the input data, developing and evolving phrases over time.
- The MIDI generation system driven by the Processing sketch does well to add complexity and interactivity and provides a simulation of musical structure to the output.

References

Winkler, T. (1998). *Composing interactive music*. 1st ed. Cambridge, Mass.: MIT Press.

Rowe, R. (1992). Interactive music systems. 1st ed. Cambridge, Mass.: MIT Press.

Boden, M. (1998). Creativity and artifidial intelligence. 1st ed. Holanda: Elsevier Science B.V.

Appendix

The functions currently included, and probability weights

- All notes up an octave. 0.5.
- All notes down an octave. 0.5
- Double all notes' duration. 0.1
- Half all notes' duration. 0.1
- Time-stretch to twice the segments' length. 0.3
- Time-stretch to half the segments' length. 0.3
- Shift notes within segment by half its length. 0.3
- Shift notes within segment by a quarter of its length. 0.3
- Transpose notes within segment by a randomly chosen amount. 0.5
- All notes up a fifth. 0.0
- All notes down a fifth. 0.0
- Spread notes evenly throughout the length of the segment. 1.0
- Copy segment to another of the same length. 0.5
- Do nothing. 1.2

The Max console provides information on executed functions during run-time.

Instructions for use

1) Connect MIDI keyboard or load MIDI file.
1) Hit record.
2) After an 8 bar count-in, record a 16 bar sequence.
3) After recording, hit the button.

4) Turn playback on.
5) Open processing and the randomDriveOSC.pde file.
6) Turn on OSC receive.
7) Sit back and enjoy, or interact with processing.

Data structure reference

note object =

```
            [ absolute time, pitch, velocity, duration, delta time ]
```

sequence object =

```
            [ [note], [note], [note], [note]… : seqLength ]
```

segment object =

```
            [ [note], [note], [note], [note]… : segLength ]
```

allSegments object =

```
            [ [segment], [segment], [segment]… : 31 ]
```

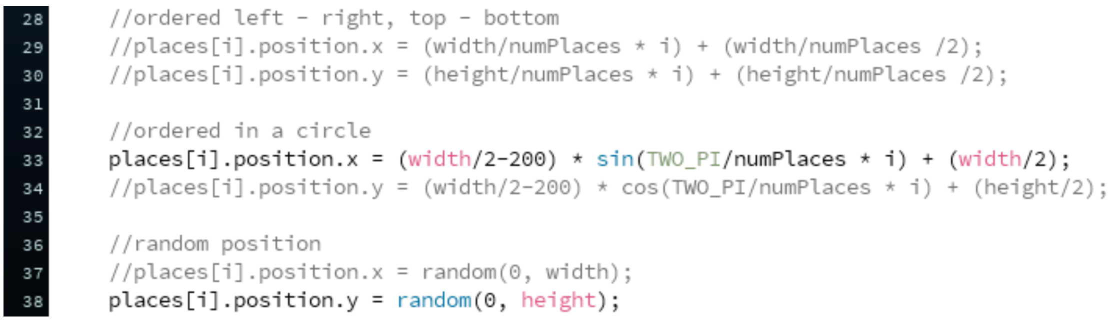
Typical Modulation Function

id = index of segment to modulate within the allSegments object.
(see Max patch for segment reference)

```
function doubleDuration(id) {
      for(var j = 0; j < allSegments[id].length; j++) { //for each
note in the segment
                  var note = allSegments[id][j]; //get the note
                  note[3] *= 2; //double the stored duration value
                  allSegments[id][j] = note; //reassign value to
object
      }
}
```

Arrangement of 'planets' in processing

The systems behavior can be affected greatly by arranging the planet objects in processing differently. Through commenting out code in this section in randomDriveOSC.pde, different arrangements can be generated. This example sets the planet's x-axis position as if they were spread around the circumference of a circle, and randomizes the y-axis position.



```
    //ordered left - right, top - bottom
    //places[i].position.x = (width/numPlaces * i) + (width/numPlaces /2);
    //places[i].position.y = (height/numPlaces * i) + (height/numPlaces /2);

    //ordered in a circle
    places[i].position.x = (width/2-200) * sin(TWO_PI/numPlaces * i) + (width/2);
    //places[i].position.y = (width/2-200) * cos(TWO_PI/numPlaces * i) + (height/2);

    //random position
    //places[i].position.x = random(0, width);
    places[i].position.y = random(0, height);
```

Included MIDI files

If a MIDI file is loaded before record is pressed, the system will simulate user input with one of the following files:

- CPentJam.mid – a simple piece in pentatonic C.
- CMaj.mid – the C major Scale.
- ChordyJam – staccato pentatonic minor chords.
- DMinArpJam – a simple piece in D minor.
- HarryPotter – the Harry Potter theme composed by John Williams