15-323/15-623 Spring 2019

Project 5. Real-Time Performance

Interim Report Due: April 11
Preview Due: April 22 & 23
Rehearsals: April 27
Concert: April 28 (afternoon)
Report Due: May 2

1 Overview

In this group or solo project, you will create a semi-autonomous performer and perform with it live in a class concert. The goal is to demonstrate mastery of many of the topics covered in class and explore deeper into the topics of algorithmic music generation and interactive control. The topics include:

1. Real-time scheduling,
2. Open Sound Control and graphical interfaces to control music generation,
3. Music representation and processing,
4. Algorithmic music generation,
5. MIDI and/or Audio processing with Serpent, Soundcool and/or MIDI synthesizers.

The "nominal" project will implement a "player," a program that receives control from a "conductor" program and, within the constraints of the conductor's instructions, generates music. Normally, a player will create only one part, e.g. drums or bass or melody or chords, etc., and the player will leave room for some real-time, high-level control by you, e.g. you might control the density of notes, general pitch range, degree of dissonance, etc. The conductor will specify tempo, downbeats, meter, style, and also selectively tell players when to play and when to "lay out." The conductor program disseminates high-level decisions from a human operator to the players in real time.

1.1 Nominal Project Requirements

In general, you should aim to address four different aspects:

1. Coordination with the other players by following the conductor (this aspect is mostly covered by the `playerbase.srp` foundation that is already provided to you),
2. Algorithmic generation of music,
3. Interactive control of music,
4. Compelling sound design, selection, and control.

In general, groups have more resources to address all of these aspects and therefore should complete more in-depth projects. In general, graduate students are also expected to address these aspects more thoroughly.

You are *not required to create a "nominal" project*. Alternatives are encouraged, including

- something creative and entirely outside the beat-based harmonic framework of the conductor/player paradigm,

- something using Soundcool, perhaps to create electronic textures that complement or contrast with the conductor and players,
- something using Max, Pd, Supercollider, or other audio processing system,
- something based on novel input devices (Wii, Camera, etc.)
- modifications to the conductor such as chord/harmony construction/sequencing,

Any of these alternatives, while encouraged, must be approved by the instructor, and we will develop a plan to include your work in the concert.

2 Interim Report (1/4 project grade)

For the interim report, you and your group (if you are in a group) should submit ONE report, with answers to the following:

- Who is in the group? (10%)
- What do you plan to create and perform? (10%)
- What is the challenge or theme that *places your project above and beyond* what you did for Projects 3 or 4? (10%)
- If the project is a team effort:
    - What roles will each person play in creating the performer? Roles include design of software components, sound design, and composition. Most group members will be working on specific software components, but achieving compelling sounds through working with sound fonts, synthesizers, MIDI parameters, Soundcool + VST plug-ins, etc. is critical (10%)
    - For each role, (1) what artifacts will be created (e.g. design documents, API specification, code, test cases, user manual, music scores, hardware), and (2) what grading criteria should be applied (e.g. "I will create an algorithm for generating bass rhythms," or "I will compose a library of riffs and encode them as Serpent data structures," etc. (10%)
    - In case this is not transparent enough, we want a rational basis to give you a good grade. Help us out by identifying something specific you can definitely do that's reasonably challenging so that at the end of the project, you can say "I did this" and we can say "Good work, here's your 'A'." Alternatively, you can say "we did not divide the work, we did not accomplish more as a group, and we did everything together" in which case we can divide 100% credit equally among team members and your grade will be 100/*n.* Please do not ask for this!

Hand in the interim report as a ***single PDF file*** via Autolab. (50%) Only one group member should hand in the report, and of course it should name other members of the group if any.

3 Preview (1/4 project grade)

Plan to have a performable version of your player by April 26. You will perform with your player for the staff (Roger, Shuqi, Caroline) on April 22 or 23 (sign up for times) so that we can give constructive feedback: The player might be great as is, or we might ask for specific changes before the concert. Our goal is to ensure that the players have sufficient technical and artistic merit to contribute to the "orchestra" in concert. If your work is not up to expectations, you'll get feedback to take corrective measures before we present to the public.

The preview performance must be well prepared and rehearsed. Your preview grade will be based on being prepared and rehearsed and meeting the criteria outlined in the interim report, e.g. meeting the challenge or theme.

4 Concert (1/4 project grade)

The concert performance should be easy since the pressure will really be on the system. Here are some tips for non-performers: This is not a demo. This is a performance. Demos are usually not fully planned and scripted. Performances usually are.

- We will set up before the concert and do a sound check. Be ready, and keep quiet while others are checking their sound.
- If you have to plug in or unplug, *be sure you are muted in the mixer or the speaker is turned off.* These speakers are *loud* and it's very unprofessional and dangerous to speakers and ears to send signals out of ungrounded or unplugged cables.
- When we play, pay attention, enjoy the performance, don't distract the audience, and don't appear disinterested. Don't perform with your lunch, drinks and snacks littering the tables of equipment.
- If you have problems in the performance, don't make faces or apologies—it's amazing what audiences do not hear, and the audience doesn't know what to expect like you do. So don't give them reasons to believe you are failing—they don't want to know that anyway.
- It should be obvious when our piece ends. Sit quietly for a few seconds at the end to let the audience enjoy the ending on their own terms. After the applause starts (and hopefully you will bring an audience!), we'll stand, face the audience and take a bow.

The concert grade will be based on being well rehearsed, prepared, and meeting the criteria outlined in the interim report. (1/2 of the concert grade)

In addition, there may be changes requested and new criteria based on the preview. Meeting any additional criteria is also 1/2 of the concert grade.

5 Report (1/4 project grade):

Create a report including all your work, Source files and Documentation. Put all this in a ***zip file*** and submit it through Autolab. (If there is a file size problem, talk to the course staff.)
Be sure that you include:

- A clear statement of what you did.
- A copy of the criteria by which your work should be evaluated.
- An assessment of your work: Did it work? Was it good? How could it be better? Were there unsolved problems you encountered?

Each group member should submit an individual report about his or her own work.
Rubric: 10 points for each of the bullet points above, 10 points for meeting your own criteria, 10 points for an implementation that works correctly with the conductor program (if applicable), 10 points for coding style and documentation, and 40 points for turning in the report.

6 Requirements Recap

This section has already been stated above, but just to say it again in a checklist, to the extent that it makes sense, your project should:

1. Follow the conductor (tempo, downbeat, scale, chord, bass, style, muted/unmuted)

2. Generate music algorithmically (templates and patterns and stored data are fine, but there should be some choices and variation/elaboration performed computationally)
3. Allow interactive control, using parameters for things such as density, complexity, pitch range, other options.
4. Show care in the selection of sounds and their combination and control. If you simply output to channel 0 with the default program, a velocity of 100, and every note has the same duration, that will show lack of care and consideration of sound.

7 The Conductor/Player Framework

Finally we get to the fun stuff. Let's write some code!

In the `p5.zip` file, you should find not only this document but also a directory tree of Serpent code. The tree has two programs you will need:

- cond/conductor.srp is the Conductor we will use in the concert. You will need this to control your player.
- player/playerdemo.srp is a working Player you can try out and also modify for your project implementation.

7.1 Operating the Conductor

Start the conductor in the cond directory using wxSerpent, e.g. `wxserpent64 conductor`

You will see lots of controls:

- BPM controls Beats per Minute (tempo)
- Beats controls Beats per Measure (meter)
- Scale creates a set of pitches suggested to the players for melodies, using mode (e.g. major, minor, pentatonic) and key (C, C#, etc.)
- Chord creates a set of pitches suggested to the players for chords, using chord type (e.g. major, dominant 7th) and key (C, C#, etc.)
- Bass suggests the root of the chord to players (bass players do not always play the root of the chord, but this could be considered the primary note for the bass player).
- Style is a misnomer, and simply selects synchronization test (style name is `sync`), and normal performance (style name is `rock`). When `sync` is selected, built-in code will play quarter notes to aid in adjusting latency so that all synthesizers play together. `rock` (normal) means play whatever your program is designed to do.
- Harmony can be `manual`, meaning that Scale, Chord, and Bass apply, or one of many chord progressions (blues, 1564, ii7v7, etc.). If a chord progression is selected, then the progression is a sequence of Scales and Chords, which are sequenced automatically and looped until another Harmony selection is made. When Harmony is not `manual`, *the Bass control functions to* transpose *the entire chord progression.* E.g. if Harmony is "blues" and Bass is D, then you get blues in the key of D.
- All of the previous controls take effect *only when you click the* COMMIT *button,* allowing you to enact multiple changes together with a click to COMMIT.
- METRO tells the conductor to play on beats to test synchronization.

In addition, the Conductor lists all players and provides "Solo" and "Mute" controls to select which players should play. In addition, each player has a "volume" adjustment that ranges from -90 to +90. You can think of this as an offset that the player will add to all of its MIDI velocities.

7.2 Operating a Player

You can start the player in the player directory by typing `wxserpent64 playerdemo demo`, where `demo` is the O2 service name of your player and also the name, which is used to store preferences in case you select a MIDI device. This defaults to `demo` but you should probably change the SERVICE initialization in the code to a unique name – *every player must be different when we all play together*.

The playerdemo program actually has no controls to speak of. You can adjust the latency to sync up with other players and you can "chat" with other players by typing into the chat window. There's also an Enable button you can use to silence your player.

7.3 Extending the Playerdemo

Playerdemo is just an extension of playerbase, mainly to implement the `play_a_measure()` function. Please read the comments in `playerdemo.srp`. Basically, when `play_a_measure()` is called, you should compute one measure of music, scheduling using `vtsched` in units of quarter notes. E.g. if you want to play 4 quarter notes in 4/4 time, call `sched_cause(i, …)`, where `i` is 0, 1, 2, and 3. Simple!

Note that `play_a_measure()` is only called when Enable is checked and the Conductor has told you to play. Initially, you are "muted" and you must click the box under column "M" in the Conductor to "unmute" the player. You'll see the Play state on the Player change from "NO" to "YES" when this happens.

Inside `play_a_measure()`, there are some variables you can use to get information from the Conductor. These variables are set 0.5 beats before the downbeat when `play_a_measure()` is called, so there should be no race condition between changing the variables and using them (unless you try to compute something on the final eighth note of the measure or later, so don't do that. You can, however, schedule computation every quarter note which would make your player more responsive to interactive controls.) The variables are:

- cur_style – a symbol (currently only 'rock' is implemented, so probably you should ignore this)
- cur_beats_per_measure – how many beats per measure (2 to 6). Be sure to test with all possible values!
- cur_bps – beats per second (tempo); note this is BPM / 60
- cur_scale – an array of 12 pitch classes, e.g. [t, nil, t, nil, t, t, nil, t, nil, t, nil, t] represents C, D, E, F, G, A, B, or the C-major scale.
- cur_chord – an array of 12 pitch classes, e.g. [t, nil, nil, nil, t, nil, nil, t, nil, nil, t, nil] represents the pitches C, E, G, Bb, or the C7 chord (C-major triad with a minor 7^{th}, also known as a dominant 7^{th} chord).
- cur_bass – an integer representing the root of a chord, e.g. 4 represents the pitch class E.
- velocity_offset – an integer from -90 to +90 that allows the conductor to tell the player to play louder or softer. The simplest way to use this is: Rather than playing velocity `v`, play velocity `vel(v)`; `vel()` is defined in `playerbase.srp` and just adds

`velocity_offset` to its argument, then clips the range to 1:127 for use as MIDI velocity. See `playerdemo.srp` for examples.

The `playerdemo` program computes a very simple bass line (alternating cur_bass and cur_bass + 7, which is normally the tonic and the fifth, but is a bad choice if cur_bass is not the tonic). It also plays the notes of the chord on the downbeat. Finally, it plays a random pitch every sixteenth note starting on beat 2, but only if the randomly selected pitch is in cur_scale, which means there will almost always be a mix of sixteenth notes and rests.

The playerdemo was intentionally created to be very simple and leave a lot to be desired musically. You should make something much more musical, but you will probably create just one part rather than three (melodic, chordal, and bass-like).

7.4 Player Interaction

It's hard to generate music automatically, taking care of both the note level and higher level compositional or improvisational levels. Your player will probably be more successful if you include some controls for density, pitch range, volume (or velocity), complexity, etc. You should add these controls. There's room for a few controls in the default window of playerdemo, but you can make the window bigger or you can open another window. You can also add OSC controls and use TouchOSC.