Impromptune

Sprint 3 Retrospective

Team 5
Ben Ahlbrand
Chris Doak
Sean Phillips
Jacob Richwine

Sprint 3 Successes

The three main goals of Sprint 3 were to finalize the GUI, test usability and fix bugs, and further develop and implement the generation options. We successfully accomplished all three goals, and feel the program is in a stable, usable, and production quality state. We are also happy about the implementation of the overall generation and generation options results, as the generated music based upon input is pleasing, dynamic, and varied enough that it doesn't produce same results every generation.

- Generation options produced different and "pleasing" results
- Addition of familiar GUI options (Recent files, prev/page)
- Fixed numerous GUI and composition bugs
- GUI was polished for a more user-friendly experience

Sprint 3 User Story Progress:

1. As a developer, I would like to have the music generation process take a maximum of a few seconds. (Complete)
2. As a user, I would like to merge generated music with original (Complete)
3. As a user, I would like to have multiple projects open at same time. (Complete)
4. As a user, I would like to choose from different parameters for algorithmic composition. (Complete)
5. As a user, I would like to have control over the generative patterns' parameters. Complete)
6. As a user, I would like a bug-free experience and user-friendly GUI (Complete)
7. As a user, I would like to be able to use a MIDI controller for manual input (if time allows) (Future work)

Overall Project Successes

We feel this project accomplished what our original design goals and ideas were. We are proud of our final results as the code is stable and easy to use, and we successfully manipulated 3rd party code to our needs. We worked well together as a team and learned a lot about successfully designing and implementing a large project. The biggest success is the generation ability. This was a feature completely foreign and new to everyone on the team, and our final version shows that the generation feature successfully analyzes the current score and produces, pleasing accompaniment to go along with the original.

Sprint 3 Challenges

Packaging

Due to the combined file structure of Zong and Impromptune, the output for native bundles had inaccurate file paths. This made packaging Impromptune much more difficult. Some possible solutions would be to find any instances of inaccurate absolute file paths in the Zong framework and change them to a relative counterpart. Another possible solution would be to create an installer and use the appdata folder for Windows machines or the equivalent folder for any other machine. With time constraints, both of these options proved to be an impractical use of time.

Expanding Generation options (affecting multiple components)

The generation options posed a challenge as each modification made would have a cascading effect across several classes, for instance training multiple files would require an overhaul of the design for holding metadata and consequently the parsers. Despite that, we managed to produce melodic accompaniment with dynamic rhythms based on the transition durations and some stylized rhythmic patterns as a component of building the inner voices in the Virtuoso class. Repetitiveness could have been more robust with better phrase planning, such as building rhythms and durations differently so as to track the different phrases and limit them accordingly. We were successful in implementing the parameters, particularly with additional voices and order. Another roadblock was coordinating changes to support arbitrary keys across several classes to cut down on generation bugs - which was also successful. The project successfully provides a user with the option for some musical brain storming along while being able to sketch it out in Impromptune or importing from another piece of software.

Overall Project Challenges

Manipulating and using the Zong! codebase was the biggest challenge we faced during this project. As the Zong! code was not created to be easily manipulated or used in other projects, we had to modify the original code sometimes and work around it so that the features we wanted to implement worked correctly. Also, the generation ability and options were a huge challenge as no one in the team had created something like that before. A lot of time went into planning this feature and deciding on how to implement it. Once that was completed, choosing which limited feature set we would support and figuring out how to generate pleasing music was the next greatest challenge.

How should we improve?

Moving forward with this project, there are several issues that need to be improved. The first one is to rework the parsing code and interrelatedness of the various components. Many times when we made a change to one class, it produced a ripple effect of bugs. As a result, we had to fix issues created by the change in many other classes. To prevent this in the future, we need to do better planning on how unique, separate components interact and pass objects around. Furthermore, more diligence is needed in picking a library to use beforehand. Zong! presented many obstacles for us, and even though it may be the only code-base available to us, we were unaware of the issues with it starting the project. We have also learned that we need to account for how changes made to one person's code may or may not affect others' code, and, as a result, that it is useful to have some knowledge of what other people are working on.