Impromptune

Sprint 2 Retrospective

Team 5
Ben Ahlbrand
Chris Doak
Sean Phillips
Jacob Richwine

1. What went well?

We consider Sprint 2 a significant milestone and major success. The GUI is looking more like a functioning project, many bugs were fixed, and major features were incorporated. Mainly, the music generation feature was successfully implemented and we are successfully analyzing and generating from a manually composed piece. We resolved Sprint 1 issues including: user deadlock and github sharing/intellij files.

- Added many GUI Features(Save as, Tabs, Undo/Redo)
- Allowed User to modify Title, Creator, Tempo, and Key of composition
- Implemented the ability to analyze currently loaded composition
- Implemented ability to generate from the analyzed music
- Load the generated, new composition into the GUI.

User Story Progress:
As a user, I would like to save and load my work. (100% Complete)
As a user, I would like to be able to compose via a GUI. (95% Complete)
As a user, I would like to be able to edit / interact with the composition 95% Complete)
As a user, I would like to add dynamics to my music(80% Complete)
As a user, I would like to have multiple projects open at same time(100 % Complete)
As a developer, I would like to have the music generation process take a maximum of a few seconds(100 % Complete)

2. What did not go well?

Many of the features implemented in this sprint had to be reworked multiple times due to not being compatible with the underlying Zong! code. Particularly, we had to rethink the undo/redo ability multiple times to successfully save the state of the current composition as over 100 classes are used in creating one note. The majority of the Sprint was also spent creating features that Zong! lacked, including an XML parser for modified files, the aforementioned undo/redo, the ability to modify a loaded composition, and a way to grab the Zong! note data for analyzing in the generator. We also ran into an issue where we would add one feature and it would require code to be reworked in many other objects.

3. How should you improve?

For the final sprint, we need to test more and more frequently. In the previous sprint, there were some bugs that impacted major features of the program that were not discovered until later. We also need to try to write code that does not affect other classes and cause a cascading effect of changes to be made.