

Music Score Visualization

Project Partner: Patrick Donnelly

Group Members:

- Kobe Chenea
- Paul Koos
- Victor Marquez
- Alex Prestwich

Document Version: 5.0

Section 1: Background

1.1 Project Audience

The Music Score Visualization project will directly benefit the Soundbendor Lab by developing a visualization module for the musicAI package that is currently under development. The musicAI package enables easy manipulation of symbolic music with artificial intelligence and machine learning. The package will eventually be released to the public through pyPi, the official third-party software repository for Python, for use by music and machine learning academics and enthusiasts as an alternative to cost prohibitive proprietary software.

1.2 Project Category/Sector

The Music Score Visualization project primarily relates to the educational sector of the economy. Within the field of education, this project will most affect the combination of machine learning/AI and music/sound. It will support the larger machine learning/AI toolkit by providing a user-friendly visual interface in the form of musical notation to allow for verification of correctness. Without this, the user would have a very difficult time confirming if the toolkit was working as intended. This visualizer will also provide a basis for future expansions to the project such as composition, editing, and playback.

1.3 Project Problem Statement

Currently there are few, if any, free and open source tools available that enable displaying symbolic music notation and most existing tools are proprietary and expensive. The goal of the project is to create a custom solution that will enable the visualization of symbolic music by drawing the notation using an existing glyph-based musical font. The module will work within the musicAI python package developed by the Soundbendor Lab.

The two main competitors that have similar offerings to this project's overall goal are [Finale](#) and [Sibelius](#). Both are paid products, with Finale being a license fee of \$299 and up. Sibelius is a yearly subscription starting at \$99/year. While these two commercial products allow you to create sheet music, they will not easily connect with machine learning and AI processes easily. The music score visualization project aims to bridge the gaps between expensive software and machine learning compatibility.

Section 2: Vision Statement

This project will result in a read-only music visualization software that will handle midi or MusicXML input and output a musical score onto a view window. This software will be made for the Soundbendor lab researchers who are in need of a visualization software for their research involving AI and music. Unlike similar music visualization software like [Finale](#) and [Sibelius](#), our application will be available to the public via PyPi for no cost. This project will also be something that Soundbendor lab can improve in the future by adding further functionality. As such, this project will allow researchers, or anyone working with midi or xml music files, to better understand the results of their musical research at a decreased cost.

2.1 Benefits and Value

The initial user base of the musicAI package, and therefore the visualization module, will be the Soundbendor Lab researchers. They will have worked on various aspects of the module and have the knowledge to apply the package to various projects. However, as the module will be released to the public, the hope is that it will receive widespread use by music machine learning researchers and enthusiasts alike. The benefit of our product is that the visualization software will be free to use and packaged with other functionality that ties it to machine learning applications.

The music score visualization package that we will be creating will save users considerable amounts of money. As stated previously, the leading competitors that offer similar functionality cost \$99 a year and more. Furthermore, they do not integrate with machine learning software that the musicAI module will provide. The visualization module and the musicAI package will streamline music artificial intelligence and machine learning processes to allow for greater efficiency and a streamlined process.

2.2 Requirements

The main project requirements are listed below. The source of these requirements is a combination of the [EECS Project Portal](#) and our group's meetings with Dr. Patrick Donnelly, our project partner.

2.2.1 Supported File Formats

2.2.1.1 Requirements

- Accept a musicXML file as input. The documentation for the musicXML standard is located [here](#).

2.2.1.2 Stretch Goals

- Accept a midi file as input. The documentation for the midi standard is located [here](#).

2.2.2 Output

2.2.2.1 Requirements

- Output is a desktop application with the musical score representation displayed.
- Move the mouse towards the edges of the window to scroll in that direction.
- No pagination - the display will allow infinite scrolling in any direction with no page breaks.
 - Included in this requirement is the ability to have an unlimited number of music staves as a possibility.

2.2.2.2 Stretch Goals

- The ability to save the music score as a PDF.
- The ability to click anywhere within the musical score to collect a slice of information.

2.2.3 Unit Testing

2.2.3.1 Requirements

- Create comprehensive suites of unit tests for code validation

2.2.4 Documentation

2.2.4.1 Requirements

- Thorough and professional documentation that conforms to the latest Python style guide.

Section 3: Success Measures and Stakeholders

3.1 Success Measures

Success Measure	What is being measured?	How is it measured?	What is the expected outcome?
Graphical user interface (GUI)	The program should create an application window that displays the sheet music rendered from the parsed musicXML file.	This is measured by ensuring an application is visible to the user. There is sheet music displayed within the application. The sheet music displayed is from the imported musicXML file.	The expected outcome is that the sheet music is displayed and perfectly matches the musicXML file.
Correct alignment and display of musical score.	The program's ability to properly vertically align all the notes in a score and place them in their proper location.	This is measured by the correctness of the score produced compared to the actual musical score and seeing if there are any vertical offsets.	The expected outcome is that the program correctly vertically aligns the symbols in a musical score and that there is no vertical offset.
User input functionality.	The program should handle arrow key input, + and - key input, and mouse cursor location as input and respond appropriately	This is measured by taking a look at how the view window changes after receiving user input and comparing that result with the expected outcome.	If the mouse cursor location is at an edge, the view window should scroll in that direction. Arrow key inputs will also cause the window to scroll in the corresponding direction. + or - key input would result in the view window zooming in or out respectively.
Advanced slicing functionality	The functionality of the slicing feature within the musical score data object.	This is measured by the correctness and level of detail in data that the slicing feature provides.	The expected outcome is a slicing feature that allows the user to "grab" any section of the musical score and store that section as a new musical data object.

Organized musical score data structure	The organized structures and substructure of the musical score data structure.	This is measured by how all of the substructures are contained in the correct location to make accessing data efficient and legible.	The expected outcome is to have all of the data structures be encapsulated in the correct locations all within the overarching musical score structure.
Automated testing and unit test coverage	The amount of code being covered by either automated testing, or unit tests.	This is measured using a Python module named Coverage .	Testing coverage of 65% or greater.

3.2 Stakeholders

There are a variety of stakeholders with differing stakes in the project. Dr. Patrick Donnelly, the project partner, not only has a stake as a project partner, but is also the leader of the musicAI python package, the larger python package that will contain this visualization module. The module will be a key functionality of the package that he will be listing publicly on PyPi to distribute to the public. Furthermore, the Soundbendor Lab has a stake as their members will continue developing the musicAI package that will utilize the visualization module. Another key stakeholder is the public final users that will be utilizing the package once it is made available to the public. They will expect a seamless integration with the rest of the musicAI package and an enjoyable user experience while utilizing the visualization features. The final stakeholder is the development team. The development team will be creating a robust, well-designed, and rigorously tested module to meet the expectations of all stakeholders.

Section 4: Project Constraints and Risk

4.1.1 Time

The project will be able to be completed within the reasonable limit of the capstone project timeline. There are no other time constraints dictating the progress of this project. We will be completing the exploratory and ideating phase by the end of the fall term, with the goal of implementation and iteration in the remaining time. The group will be able to work within the reasonable expectations for hours per week set forward by the instructors. Time will not be a significant limitation when considering the work required to complete the project.

4.2.1 Resources

This project will make use of a variety of free resources available to us as well as some music notation program licenses that would need to be purchased. In terms of implementation, this project will make use of Pyglet and the SMuFL compliant font Bravura at no additional cost. As of our most recent meeting, it does not seem likely that we will need to purchase any additional implementation resources. That being said, it does seem likely that we will need to purchase a license for a musical notation program. This is needed to test our software implementation as well as developing MusicXML files from musical scores.

4.3.1 Scope

For the music visualization feature our minimum features will include a fully visualized musical score sheet that allows the user to navigate through via scrolling. It will also include a slicing feature capable of taking portions of the musical sheet and capturing them as their own virtual objects. Possible features include more UI features to the visualization such as zooming, capturing portions of the music with a selection box and exporting the musical file as a PDF. The main constraint with this project is time as most of the development process for this project comes from planning and implementing software solutions.

















































4.4.1 Risk Management

Risk ID	Risk Description	Risk Category	Risk probability	Risk Impact	Performance Indicator	Responsible Party	Action Plan
R1	No work over Winter break	Timeline	50%	M	Lack of GitHub commits last half of December	Paul	Retain
R2	Typesetting license becomes proprietary	Technical	0%	H	Email from or post on www.smufl.org	Victor	Avoid
R3	Performance issues with long musical scores	Technical	25%	L	Application unusable after importing file of certain size	Victor	Avoid
R4	Project disagreements within the team	Team	25%	L	Difficulty making project decisions	Kobe	Retain
R5	Feature	Timeline	75%	M	Moving optional	Paul	Transfer

	Creep				features to requirements		
R6	Not completing project requirements on time	Timeline	50%	L	Missed deadlines	Alex	Reduce
R7	A team member drops from the project	Timeline	0%	H	Communication from team member	Kobe	Transfer
R8	Project Partner cancels the project	Timeline	0%	H	Communication from project partner	Alex	Transfer
R9	Loss of code repository	Technical	0%	H	No access to code on GitHub	Alex	Transfer
R10	Personal Vacations	Timeline	75%	L	Communication from Team Member	Victor	Retain
R11	Missing a weekly project meeting	Timeline	75%	L	Communication from team member	Kobe	Retain
R12	Project bugs	Technical	100%	L	Application not working as expected	Paul	Retain
R13	Pyglet determined incompatible with project requirements	Technical	10%	H	Team consensus requirement cannot be met	Victor	Avoid
R14	Unsupported Python functionality	Technical	50%	L	Stated work is not being completed as planned	Kobe	Retain
R15	Complete project requirements early	Timeline	50%	L	Requirements are verified complete by team and project partner	Victor	Transfer
R16	Group loses	Team	20%	L	No communication	Kobe	Reduce`

	communicati on with team member				with team member after multiple attempts		
R17	Project transition to commercial product	Technical	10%	H	Project partner tells us this is now the intention	Alex	Retain
R18	Cannot find a solution to specific project requirement	Technical	5%	M	Lack of progress towards a given requirement for 3 weeks in a row	Alex	Avoid
R19	No unit or automated testing	Technical	50%	M	Requirements being fulfilled and no tests being written towards them	Paul	Reduce
R20	Unit or automated testing is failing	Technical	20%	H	Unit test results are being ignored while work is being done	Paul	Reduce

Section 6: Iteration Plan

Gantt Chart (Weeks)																				
	Alex (TM1) 				Kobe (TM2) 				Paul (TM3) 				Victor (TM4) 				Everyone (ALL) 			
	Fall Term				Winter Term										Spring Term					
Task	7	8	9	10	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	
Score Window Class																				
Glyph Class																				
Glyph Placement Algo. Dev																				
Glyph Algo. Testing																				
Glyph Algo. Revisions																				
Spike - absolute boundaries																				
Scrolling Event Handlers																				
Other Event Handlers																				
Zooming Event Handlers																				
Data structure to hold music stuff																				
Musical Score slicing																				
Custom slicing of data structure																				
Module Integration Testing																				
Beta testing (Goal)																				
Project pass-off (Goal)																			