Sheet Music Transcriber

Company 1

HappyJam

Team Introduction

Company 1

Matthew Marinets	CEO and Software Engineer	System architect, OpenGL rendering
Jaskirat Arora	CCO and Systems Engineer	Documentation and Planning
Haoran “Harry” Hu	Software Engineer	UI development expert
Polina Bychkova	Software Engineer	Algorithms development
Akaash Parajulee	Electronics Engineer	Testing Framework, MIDI Export
Avital Vetshchaizer	Software Engineer	Algorithms development

Overview

Background

- Music is a sound, it is useful to have a way to write it down and store it – sheet music, MIDI
- Performing music is straightforward, transcribing sheet music is harder to learn and time-consuming to do
- Requires a trained ear and repeated listening
- The high barrier to entry makes it difficult for learning musicians to learn all the songs they may want to play
- Sheet Music Transcriber is a solution to this problem

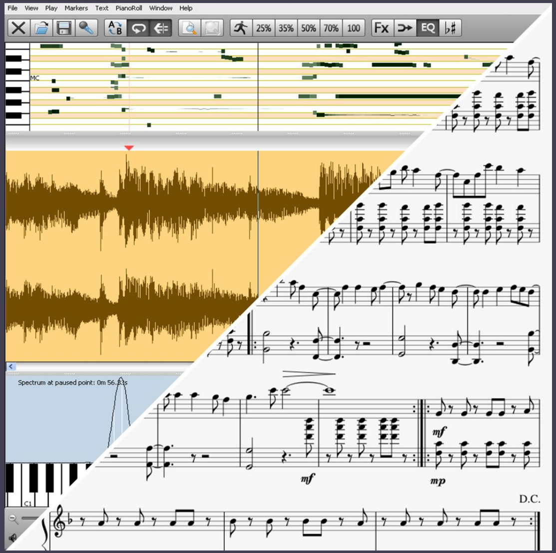


Figure 1: Music Transcription [1]

Background — The structure of sound

- Sound is changing air pressure
- Digital sound is a list of numbers representing air pressure measurements taken at a regular frequency
  - Every-day sampling rates may be 44.1 kHz (CD quality), or 48 kHz
- Sound is a signal, and so we can get its frequency spectrum
- We only want frequency information for small slices of time, so we use the Short-Time Fourier Transform (STFT)
- By lining up many STFT spectra for ordered, adjacent (or overlapping regions), we may make a spectrogram

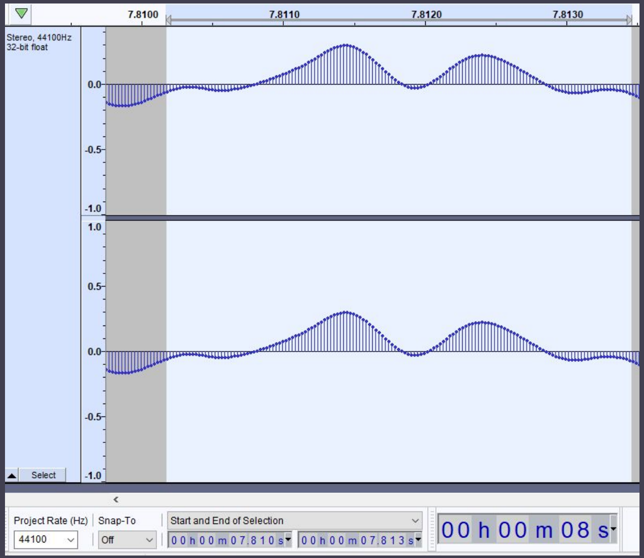


Figure 2: A zoom-in of audio in Audacity, showing individual samples

Background — Spectrograms

- A spectrogram is a time-frequency representation of a signal
- It lets us see the short-time frequency content of a song and how that changes with time
- Reveals the structure of notes and periodic sound – a collection of parallel, horizontal, equally spaced lines

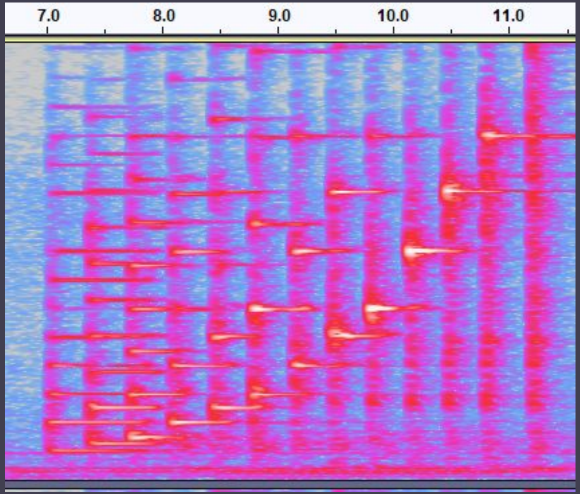
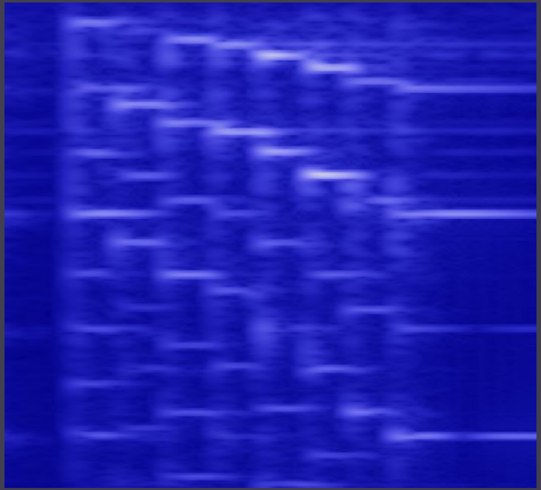


Figure 3 (Left): a spectrogram in Audacity (lower frequencies below)

Figure 4 (Right): A portion of the spectrogram in SMT (lower frequencies above)



Motivation

- A couple of our members have musical knowledge
  - One Such member identified an opportunity
  - Other members were interested in learning about music
- Market research
  - English speaking musicians
  - Communities in this transcription area online
  - More later in the presentation

Motivation

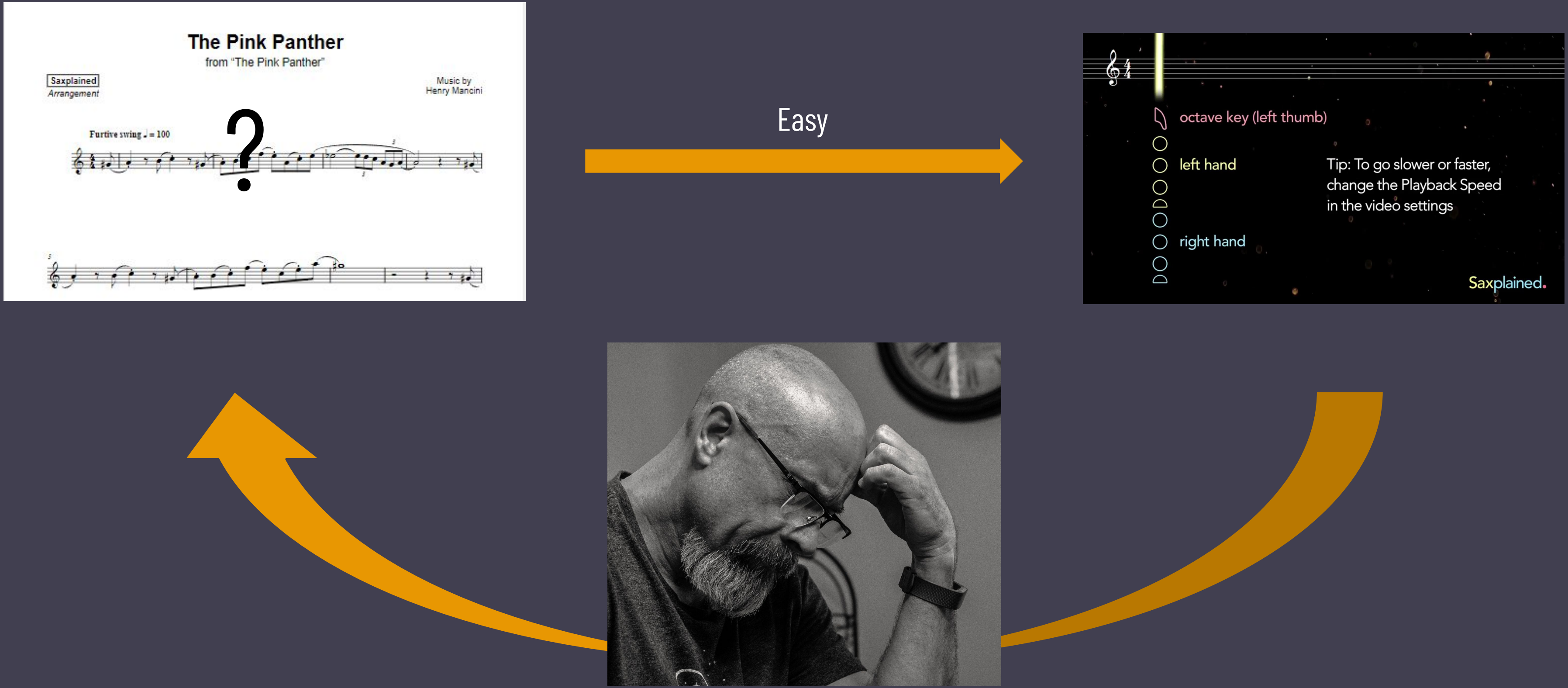


Figure 5: Music Transcription [2][3][4]

Technical Case — UI (Qt)

- Pyside 6 (Python) and Qt are used to implement the user interface
- ui_main.ui, ui_main.py and mainwindow.py are used
- Hierarchical structure

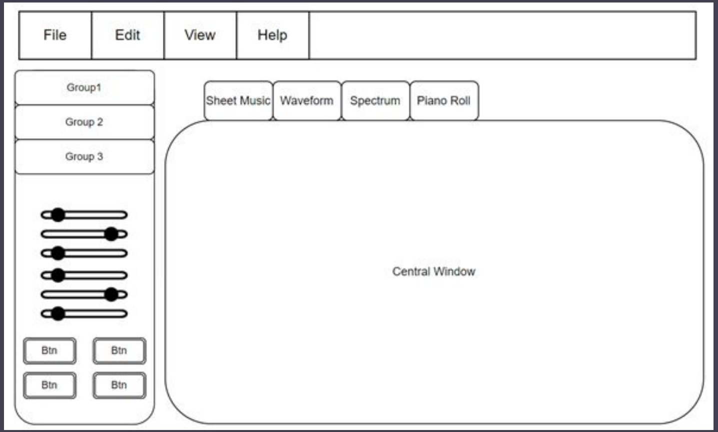


Figure 6: The aimed UI design for SMT

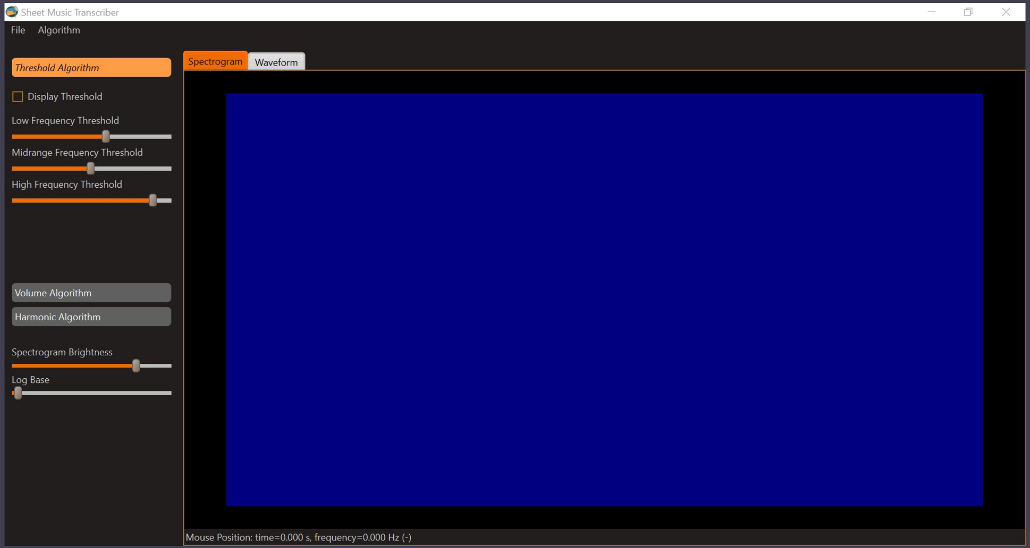


Figure 7: The achieved UI design for SMT

Technical Case - UI (OpenGL)

- Pure Qt was too slow and inflexible for rendering the spectrogram, so we turned to OpenGL
- OpenGL is a hardware graphics API for GPU programming and high-performance graphics
- This allows us to render large spectrograms and perform operations on them in realtime
  - Warping for log-scale
  - Per-pixel operations for brightness adjustment
  - GPU instancing for quickly adding notes to the display

Video 1: UI based on OpenGL

Algorithms - Variable Threshold

- The simplest algorithm – it outputs notes wherever the PSD (Power Spectral Density – spectrogram intensity) is above a configurable threshold and on a 12TET note frequency
- SMT provides dynamic visualization for what's above the threshold
- Note harmonics are not filtered

Video 2: Variable Threshold

Algorithms - Harmonic Elimination

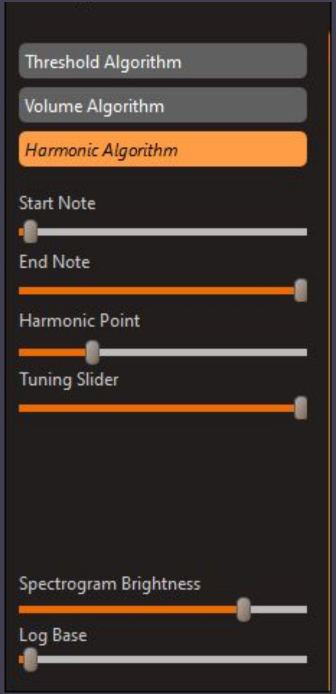


Figure 8: Harmonic Sliders

- Feeds algorithms with Start and Notes for note detection
- Feeds algorithm with Harmonic point to remove harmonics
- Tuning Slider adjusts algorithm sliders

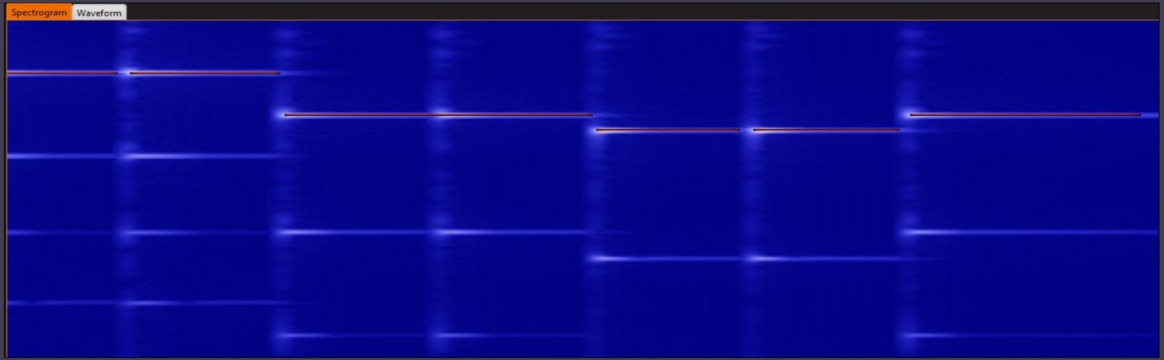


Figure 9: Sample song note detection

Algorithms - Volume Based detection

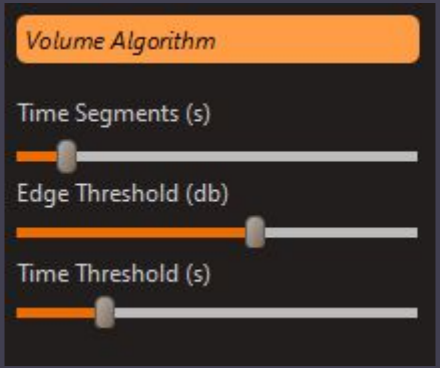


Figure 10: Volume Parameters

The Volume-based algorithm takes the “loudest” note at any given windowed sample and assigns it a note. There are a few parameters that control its behaviour:

Time Segments - Controls the length of each sample taken by algorithm. It can change from 1s to 10s.

Edge Threshold - Defined as the minimum required threshold for the next sample in the next sample to be considered a new note. If the next sample does not have pass the threshold, that sample is considered part of the same note. It is defined from 0.1db to 5db.

Time Threshold - The minimum time between notes. Defined from 0s to 20s.

Technical case – Architecture

- “Architecture” refers to the macro-level structure of the program, the parts that are hard to change
- Software architecture is a service mainly felt by developers. It’s main effect on end-users is in performance and responsiveness
- The SMT has a structure of different modules running on different threads and communicating with each other using messages
- Multithreading segregates view components from the algorithms and file I/O, meaning the interface stays responsive even when running slow tasks
- Messages are simple to define and send, easing development
- The messaging structure provides an easy framework to hook into for other forms of input; we use this for a command-line interface for automation and power-users

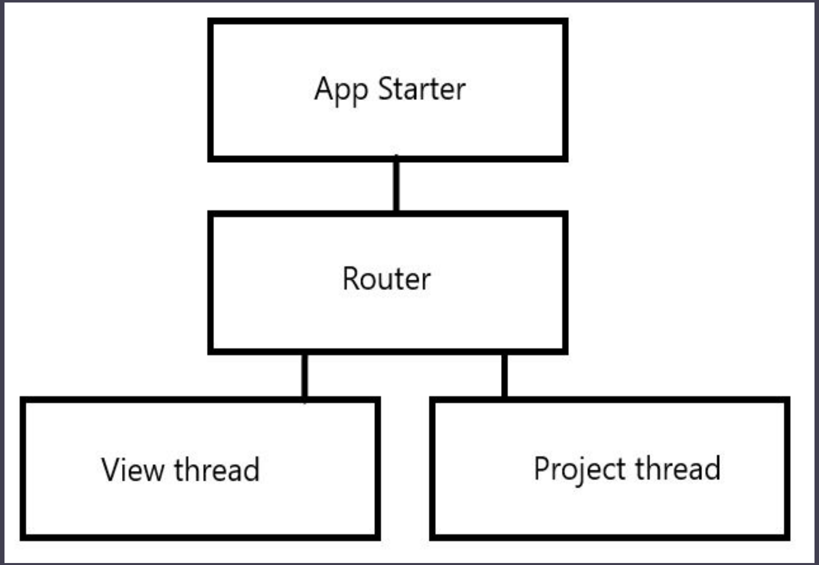


Figure 11: System Architecture

Set Up and libraries

Libraries are downloaded locally by first running SetUp.bat.
Our libraries are the software equivalent of materials.

The following libraries were used during the development of our software:
numpy - Matrix Arithmetics
pyside6 - Provides access to the complete Qt 6.0+ framework.
Scipy - Provides algorithms for optimization, algebraic equations, and many other classes of problems.
PyOpenGL and PyOpenGL_accelerate - Provides rendering tools
Audioread - Converts audio files to samples
Mido - Converts a list of notes to a MIDI file

The Music Transcriber uses libraries from the FFmpeg project under the LGPLv2.1

Figure 12: Dependencies used

Recording Equipment Test Results

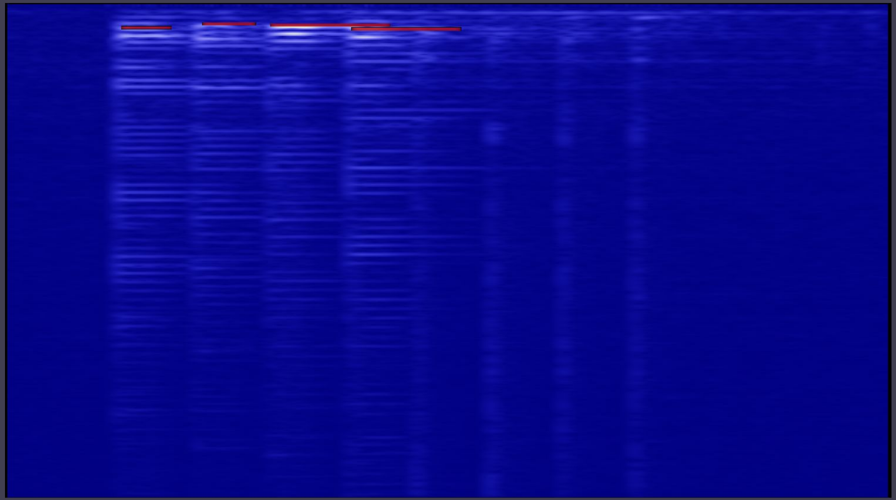


Figure 13: Bidirectional

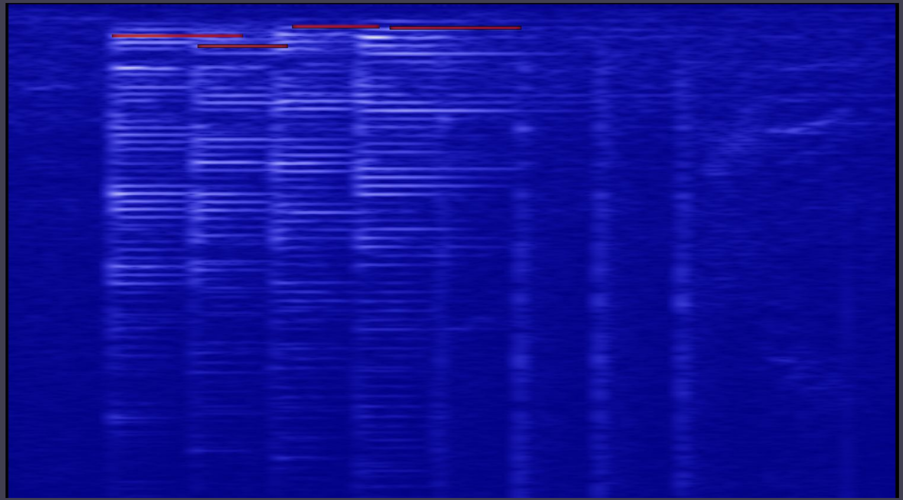


Figure 14: Omniidirectional

Testing Methodology

- How to measure?
  - Note Onset
  - Note Offset
  - Note List Length
  - True Note Comparisons

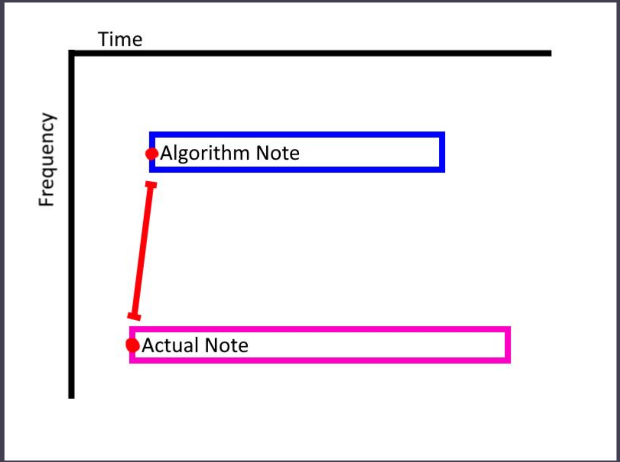


Figure 15: Onset distance

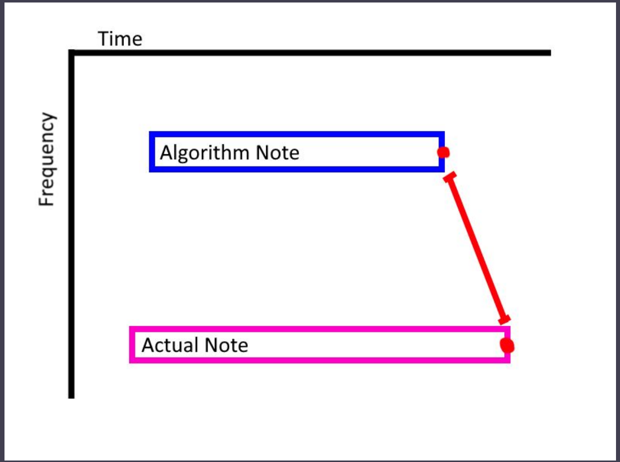


Figure 16: Offset distance

Testing Methodology

- How to measure?
  - Note Onset
  - Note Offset
  - Note List Length
  - True Note Comparisons

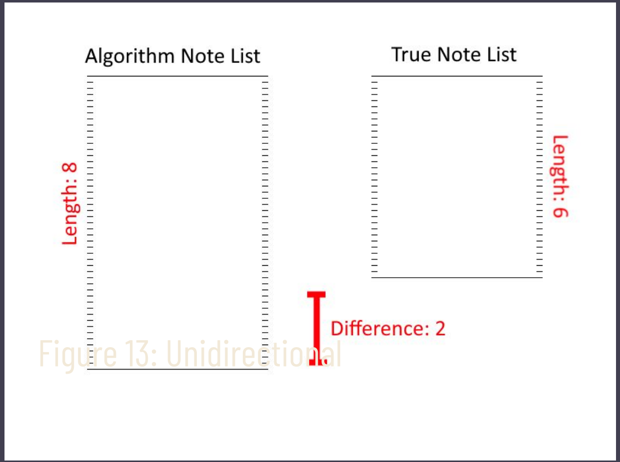


Figure 17: Note List Length

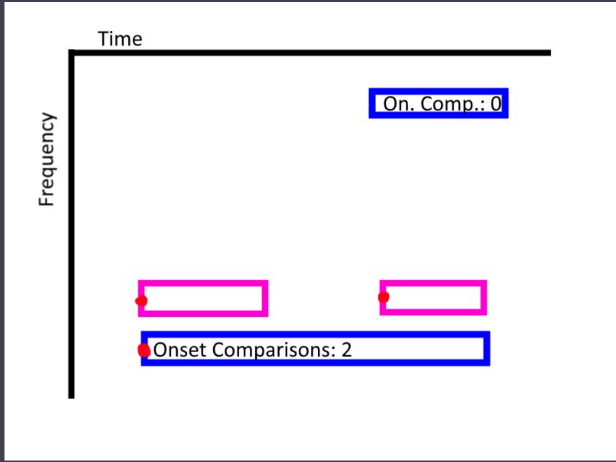


Figure 18: True Note Comparisons

Example Testing Results

	Algorithm: Volume				Test: A4toE5-8quarterNotes			
	Parameters				Test Results			
	Time_segments	Edge_threshold	Time_threshold		Note List Length Diff	Avg Onset Dist	Avg Offset Dist	Highest Comparison
Default	2	3.2	5		11	3.087558652	29.80585858	3
	1	3.2	5		14	3.198422673	30.87895483	3
	2	3.2	5		11	3.087558652	29.80585858	3
	3	3.2	5		10	3.18865896	30.75717324	3
	4	3.2	5		6	2.946973115	28.43208296	2
	5	3.2	5		5	3.016112222	29.05765239	2
	6	3.2	5		5	3.290721461	31.69397803	2
	7	3.2	5		2	3.175466573	30.57360557	2
	8	3.2	5		2	3.189259041	30.69059908	2
	9	3.2	5		3	3.112168289	29.97198033	2
	10	3.2	5		2	2.672735619	25.71063014	2

	Algorithm: Volume				Test: A4toE5-8quarterNotes			
	Parameters				Test Results			
	Time_segments	Edge_threshold	Time_threshold		Note List Length Diff	Avg Onset Dist	Avg Offset Dist	Highest Comparison
	2	0	5		141	3.229178198	30.84246386	20
	2	0.5	5		125	3.248626304	31.07058202	18
	2	1	5		69	3.309576034	31.76911405	12
	2	1.5	5		18	3.176315711	30.64166501	4
	2	2	5		14	3.205475639	30.93878106	3
	2	2.5	5		13	3.239240212	31.26434353	3
	2	3	5		12	3.101332902	29.93809891	3
	2	3.5	5		11	3.087558652	29.80585858	3
	2	4	5		10	3.218606025	31.06662322	3
	2	4.5	5		9	3.159132184	30.49410926	3
	2	5	5		9	3.159132184	30.49410926	3

	Algorithm: Volume				Test: A4toE5-8quarterNotes			
	Parameters				Test Results			
	Time_segments	Edge_threshold	Time_threshold		Note List Length Diff	Avg Onset Dist	Avg Offset Dist	Highest Comparison
	2	3.2	0		11	3.114780556	30.0367668	3
	2	3.2	2		11	3.114780556	30.0367668	3
	2	3.2	4		11	3.102076992	29.92900963	3
	2	3.2	6		11	3.087558652	29.80585858	3
	2	3.2	8		11	3.087558652	29.80585858	3
	2	3.2	10		11	3.087558652	29.80585858	3
	2	3.2	12		11	3.087558652	29.80585858	3
	2	3.2	14		11	3.087558652	29.80585858	3
	2	3.2	16		11	3.087558652	29.80585858	3
	2	3.2	18		11	3.087558652	29.80585858	3
	2	3.2	20		11	3.087558652	29.80585858	3
Max	10	5	20		2	2.638254654	25.4181464	2

Figure 19: Testing Examples

Schedule - expected

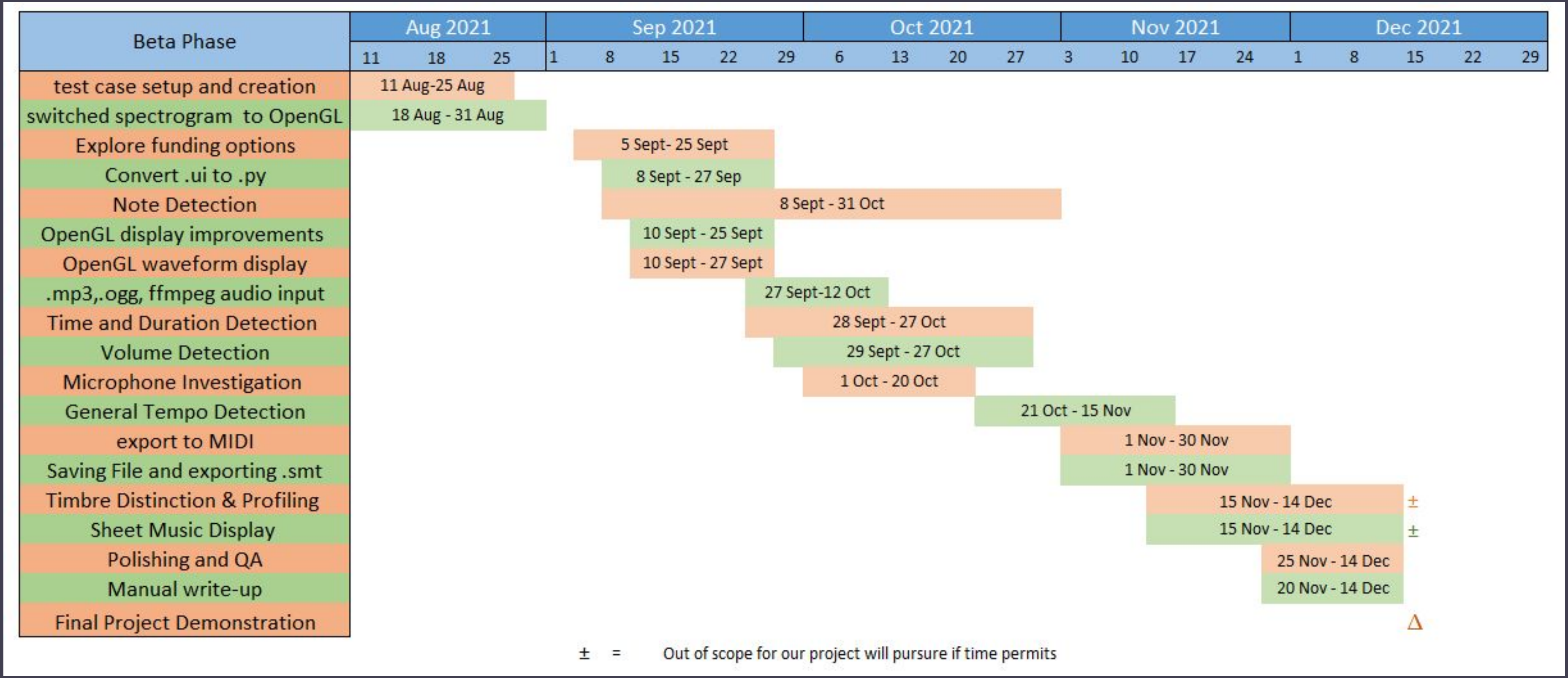


Figure 20: Schedule at beginning of Beta Phase

Schedule - actual

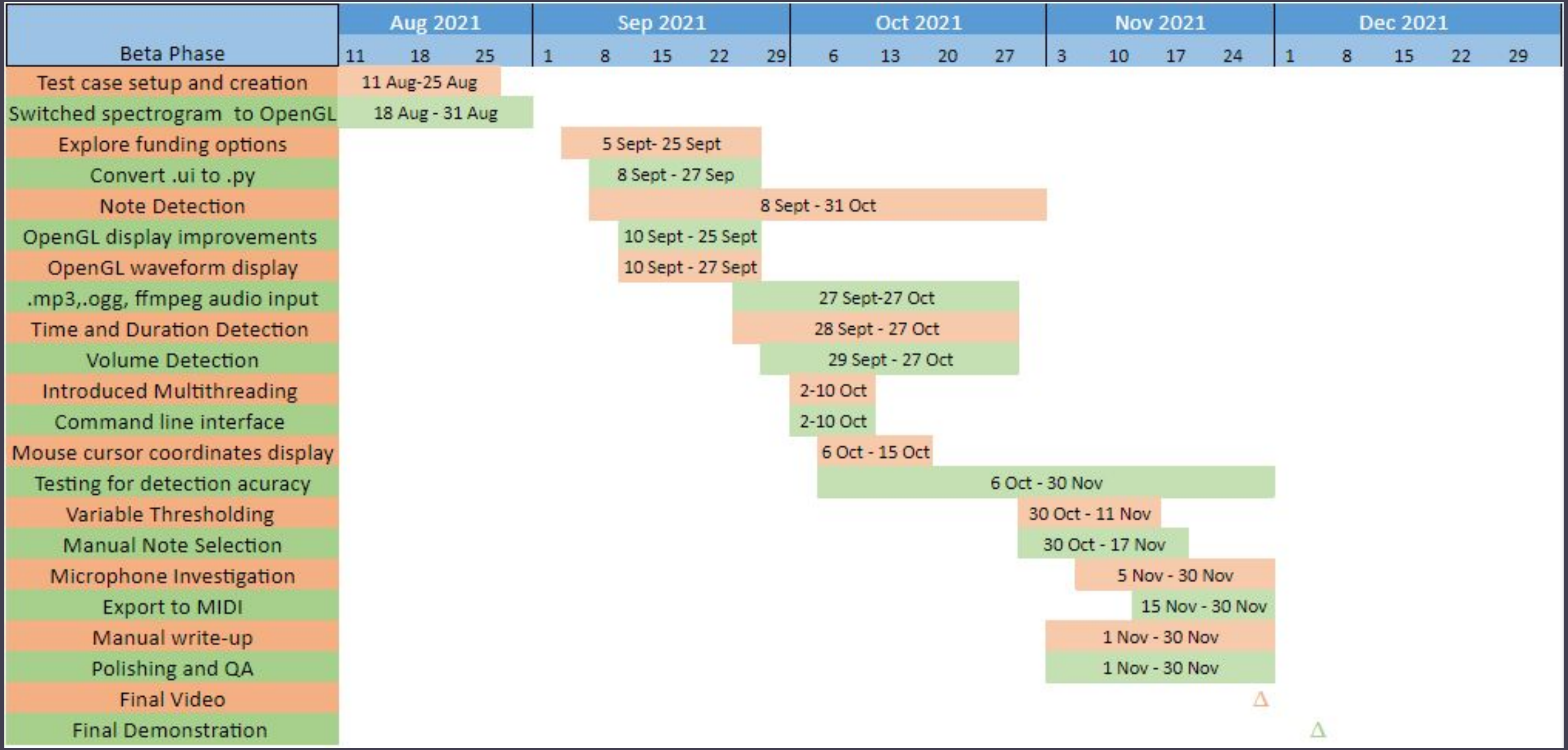


Figure 21: Schedule at end of Beta Phase

Business Case - Market Description

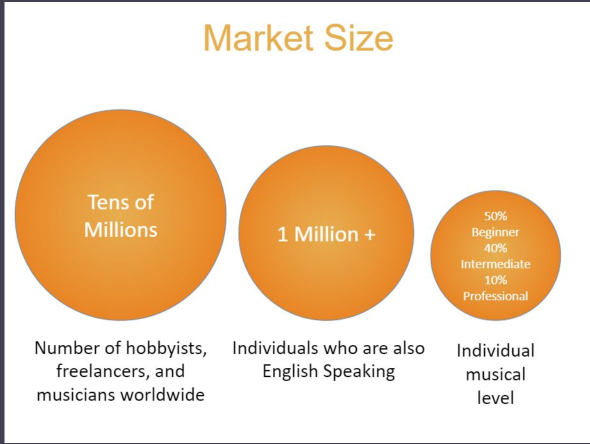


Figure 22: Market SIze

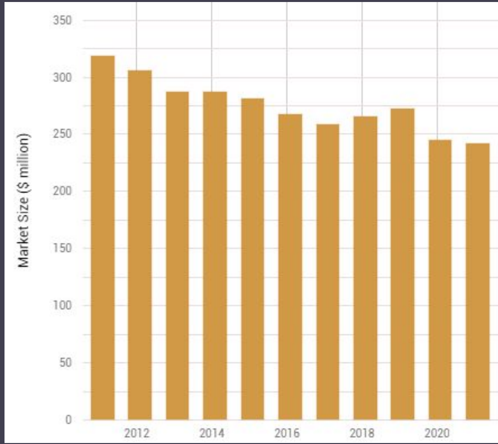


Figure 23: Sheet Music Publishers in the US [5]

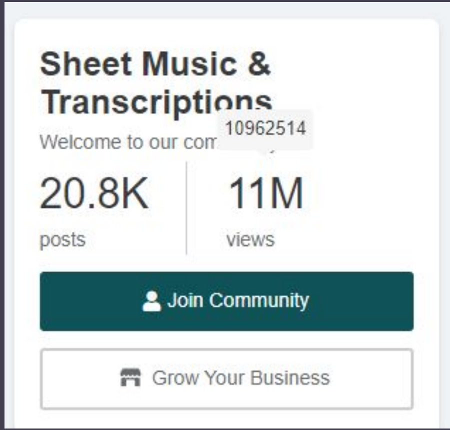


Figure 24: Transcription Forum [6]

Business Case - Competitors

List of Competitors:

- ScoreCloud
  - Automated
  - $19.99 / Month
- TransribeMe!
  - +2M members
  - $47/h
- Upwork
  - Hundreds of hobbyist members
  - ~$20/h

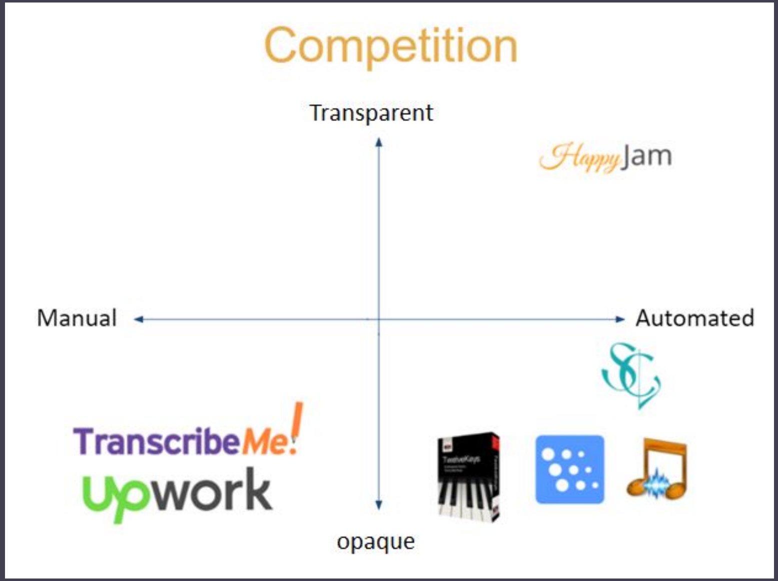


Figure 25: Competitor Graph

Business Case - Financials

- Sheet Music Transcriber as a monthly subscription of $9.99/month

- Operation Costs
  - $140/year for domain acquisition and maintenance
  - $360,000/year in salaries
  - $76,117/year for marketing
  - $12,000/year for company meetings and miscellaneous
  - $448,257/year for operations or $37,355/month

- To break even 3740 active subscriptions every month

- Initial investment of $225,000 to smoothly run operations for 6 months

Risks Management

- No health and safety risks for both users and developers
- Certain Risks associated with development and how to overcome them
  - How to know our algorithm's performance
  - What if one algorithm is queued right after another
  - What if a user wants to input a non supported file
  - What if our algorithm misses a note or detects something wrong
  - What if a user selects a note out of bounds of the display
- Plan B for commercialisation - target music institutions and instructors (1,795 institutions is USA have degree-granting music programs[7])

Engineering Standards

- **ISO/IEC 23000-12:2010 -** Information technology — Multimedia application format (MPEG-A) — Part 12: Interactive music application format [8]
- **ISO/IEC 24752-8:2018 -** Information technology — User interfaces — Universal remote console — Part 8: User interface resource framework [9]
- **ISO/IEC 29138-1:2018 -** Information technology — User interface accessibility — Part 1: User accessibility needs [10]

Self Reflection

- Overly-large scope at the beginning of capstone
  - We dropped timbre analysis given the difficulties associated with it
  - We switched from sheet music export to MIDI
  - Multiple notes at the same time wasn't consistently doable
- Feedback from meetings
  - Automated testing – getting a numeric metric of algorithms
  - Axes, legend – not enough time to do
  - Market analysis – find more accurate numbers for the demand for transcriptions
- We decided on a project idea before fully understanding everyone's technical competencies
  - If we did this again, we would have to communicate this more clearly, both offering more precise information and being more specific in requesting information
  - We needed to commit to workflow standards earlier to avoid workflow conflicts

Self Reflection — what did we learn?

- More frequent meetings were a big help, for communication and ramp-up

And we're all proud of:

- Akaash learned Python
- Matthew implemented OpenGL GPU instancing for the first time
- Haoran learned Qt and Python
- Avital learned music, both how to play and how musical sound is structured
- Polina learned music theory, Python
- Jaskirat learned how to write interrelated business documents

Conclusion

Summary:

- Created a functional UI
- Implemented three different algorithms
- Successfully detected notes and transferred them into midi file

Future Plan:

- Detect chords
- Detect multi-instrument music
- Generate sheet music

Acknowledgements

- We would like to thank Rodney for his insights and suggestions and taking out time to meet with us and discuss our ideas.

- We would like to thank ENSC405 instructional team for their support towards this project

- We would like to thank ENSC440 instructional staff for their feedback and suggestions

References

[1] "8 BEST MUSIC TRANSCRIPTION SOFTWARE IN 2021," fixthephoto.com, Sept. 26, 2021. [online] Available: https://fixthephoto.com/best-music-transcription-software.html

[2]"Grayscale photo of a man thinking in front of an analog clock", pexels.com [online] Available: https://www.pexels.com/photo/grayscale-photo-of-man-thinking-in-front-of-analog-wall-clock-1194196/

[3]"The Pink Panther", musicnotes.com [online]. Available:https://www.musicnotes.com/sheetmusic/mtd.asp?ppn=MN0243481&id=&ca=0

[4]"Pink Panther Sax Tutorial", youtube.com, January 14,2021 [online] Available: https://youtu.be/wP2F5ingiuQ

[5]"Sheet Music Publishers in the US", ibisworld.com, November 10,2021 [online] Available: :https://www.ibisworld.com/industry-statistics/market-size/sheet-music-publishers-united-states/

[6]"Sheet Music & Transcription", saxontheweb.net [online] Available: https://www.saxontheweb.net/forums/sheet-music-transcriptions.53/?order=reply_count&direction=desc

[7] "Information technology — Multimedia application format (MPEG-A) — Part 12: Interactive music application format," *ISO*. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:23000:-12:ed-1:v1:en

[8] "Information technology — User interfaces — Universal remote console — Part 8: User interface resource framework," *ISO*. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:24752:-8:ed-1:v1:en.

[9] "Information technology — User interface accessibility — Part 1: User accessibility needs," *ISO*. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:29138:-1:ed-1:v1:en

[10] "Facts and Figures Concerning Music and Higher Education In the United States", music.org, January 25, 2015, [online] Available: https://www.music.org/pdf/mihe/facts.pdf

Happy Jam

Questions?

Thank you and have a Happy day!