Wrock

Julia Cooper, Marilyn Sun, Gabriella Bova

The Project
Create a soundtrack for a story given as input, based on its changing sentimental tone.

Minimum Deliverable

- Program is provided with hardcoded short musical pieces for different moods (Neutral, Happy, Sad, Angry, Excited) and riffs for different characters
- The program analyzes the text by paragraph. It figures out the emotion of each section, and adds character riffs for characters who are present in that section (connected with that section's emotion)
- Then, the program takes the given music and pieces it together to create a continuous soundtrack that can be played in the background of reading the story

Maximum Deliverable

- Program can analyze a long story with many characters, like Harry Potter, and create a song on the fly to play in real time.
- Program is not supplied with predetermined musical pieces for different moods, but generates the music from scratch based on the tone of each section, using multiple instruments.
- Program can save the generated soundtrack in the form of a piece of music file.
- Program can create smooth transitions between moods

What's the biggest problem you foresee or question you need to answer to get started?
Coordinating multiple threads for playing the music the program has created
Matching tones of characters and background during analysis
Generating a soundtrack that sounds good

What have you done so far, and how do you plan to proceed?
What we have done so far

1. We chose a method for analyzing the tone of the text. We have decided to use Watson Tone Analyzer, and API provided by IBM.
2. We chose a Python music library, pyo, to use for soundtrack generation and playing.
3. We experimented with creating random melodies, both with a major and minor scale, and combining multiple melodies in harmony.
4. We have two prototypes: one program can open a file and analyze the tone of each paragraph of text in that file; the other program can play a tune in Python code.
5. Collected music samples.

Next Steps

1. Create a thread manager to coordinate concurrent text analysis and enforce an order.
2. Parse each paragraph and check for appearance of characters.

How will the work be divided among the members of the team?

We will have two groups, one working on the tone analyzer module and one on music generation at first. This is more related to learning how to use the IBM Watson and Pyo library, and less related to the overall architecture of the program and concurrency, so we are comfortable with dividing this work up.

Once these two modules are complete, we will come together to coordinate and implement the concurrent threads. Because we separately worked on different modules, we will be able to provide unique input as to how to connect and each module. But any interesting concurrency related design decision would be discussed between all three of us, since it is important that we are all familiar with how the threads connect the modules.

Interesting design decisions:

The problem we want to address

We wanted to figure out how to use concurrent threads to maximize efficiency in parsing text input and coordinating music. We could do everything sequentially, but that would be very slow.

Ways we considered addressing it

Initially, we considered using different threads for each layer of music that we add on. For example, as the program detects changes in sentimental tone, it would start a new thread, tweak a current thread, or slowly kill an existing thread. This allows for a smoother transition in music. We also considered using concurrent threads for memory storage, specifically writing different pieces of music to the same file. We thought that it might be convenient to have the concurrent threads be responsible for saving their own output. However, neither approach really fully improved program efficiency, and concurrently saving file output later proved to be difficult to do with the pyo library.

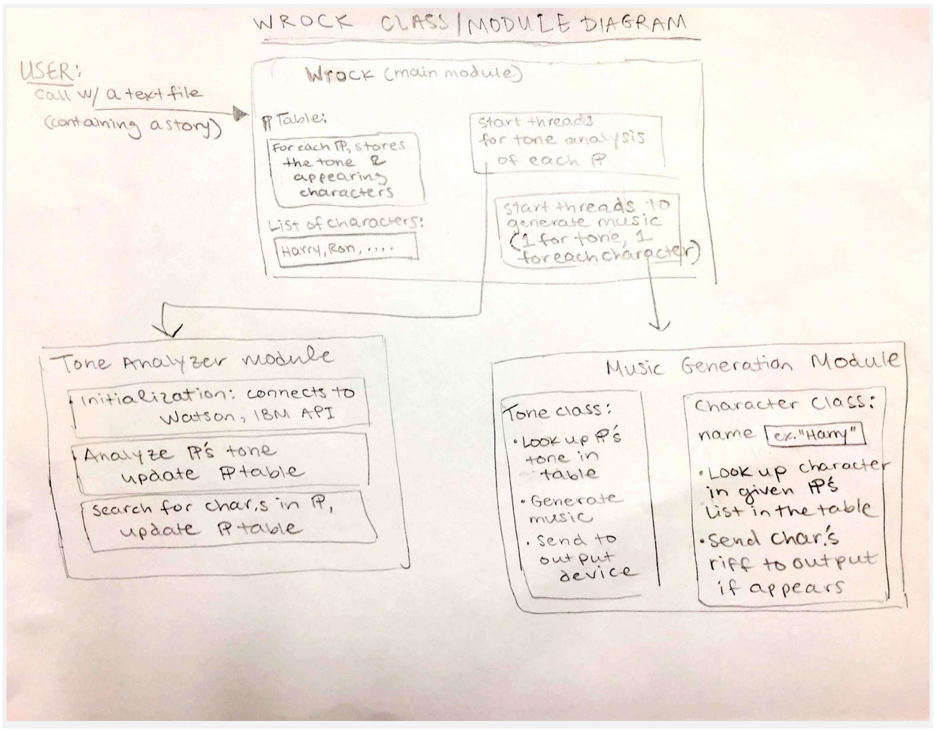
How we chose to implement and why

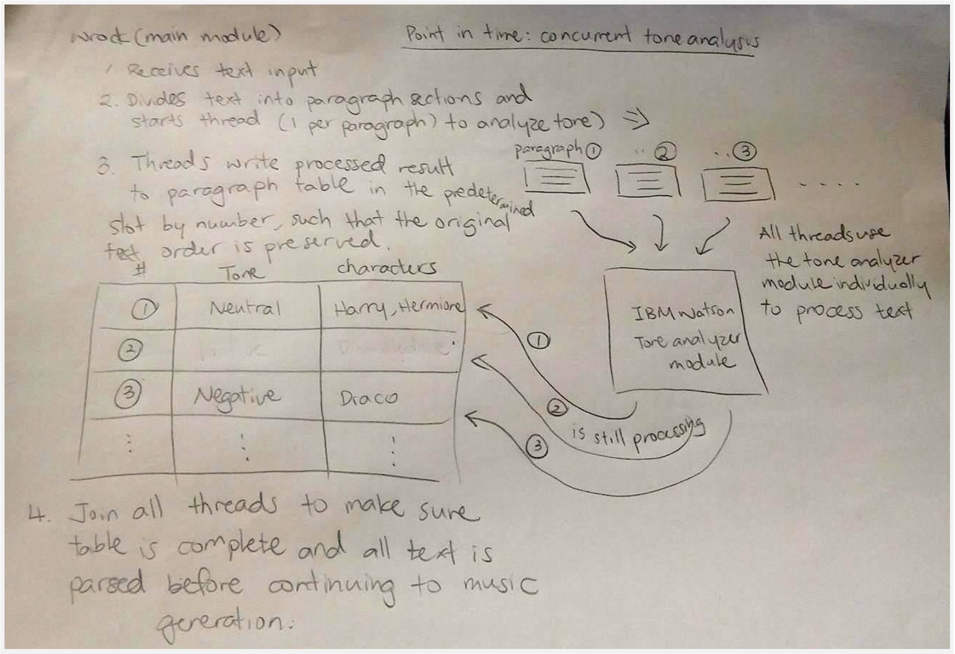
Currently, we chose the approach of using concurrent threads to play music, as suggested earlier. In addition, we decided to use concurrent threads to parse input, sectioning off different pieces text for different threads to process at the same time. This should further improve program runtime.

Timeline:

Date	Task
Mon 11.06	**Initial Design Due**
Wed 11.08	Continue experimenting with libraries Continue experimenting with thread ordering for text analysis
Mon 11.13	Finalize music threads
Wed 11.15	Experiment with playing music concurrently Connect tone analysis and music-creating modules
Mon 11.20	**Refined Design due**
Wed 11.22	Implement main module to coordinate all separate modules
Mon 12.04	Debug and test
Wed 12.06	**Team Presentations**
Fri 12.08	**Final Report due**

Diagrams:


WROCK CLASS|MODULE DIAGRAM
USER: call w/ a text file (containing a story)
Wrock (main module)
¶ Table:
For each ¶, stores the tone & appearing characters
List of characters:
Harry, Ron, ...
Start threads for tone analysis of each ¶
start threads to generate music (1 for tone, 1 for each character)
Tone Analyzer module
Initialization: connects to Watson, IBM API
Analyze ¶'s tone update ¶table
Search for char.s in ¶, update ¶ table
Music Generation Module
Tone class:
• Look up ¶'s tone in table
• Generate music
• Send to output device
Character class:
name
ex. "Harry"
• Look up character in given ¶'s list in the table
• Send char.'s riff to output if appears



wrock (main module)

Point in time: concurent tone analysus

1. Receives text input
2. Divides text into paragraph sections and starts thread (1 per paragraph) to analyze tore) ⇒
3. Threads write processed result to paragraph table in the predetermined slot by number, such that the original text order is preserved.

#	Tone	characters
1	Neutral	Harry, Hermione
2		
3	Negative	Draco
⋮	⋮	⋮

4. Join all threads to make sure table is complete and all text is parsed before continuing to music generation.

paragraph ①
.. ②
.. ③
All threads use the tone analyzer module individually to process text
IBM Watson Tone analyzer module
①
②
is still processing
③