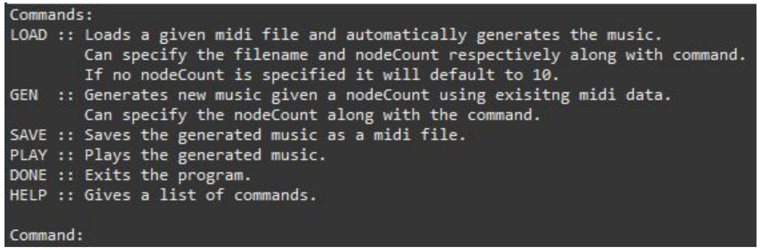
Procedural Music

Group Members: Mckade, Robert, Taylor

Review

- Generate music using an algorithmic approach.
- Get MIDI sample
- Extract features
- Use Wave Function Collapse to generate music

- Prototype
  - Used only single fixed notes



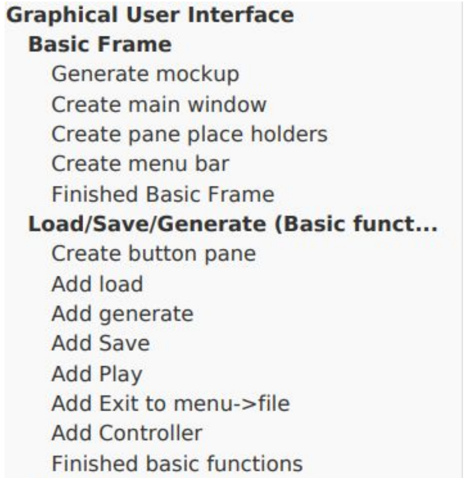
```
Commands:
LOAD :: Loads a given midi file and automatically generates the music.
        Can specify the filename and nodeCount respectively along with command.
        If no nodeCount is specified it will default to 10.
GEN  :: Generates new music given a nodeCount using exisitng midi data.
        Can specify the nodeCount along with the command.
SAVE :: Saves the generated music as a midi file.
PLAY :: Plays the generated music.
DONE :: Exits the program.
HELP :: Gives a list of commands.

Command:
```

Current Milestones/Goals

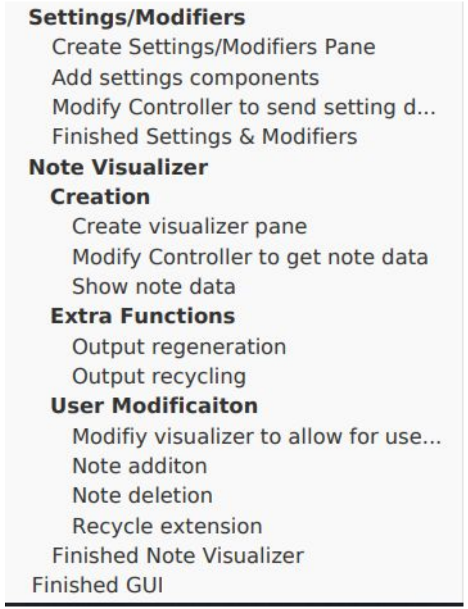
- GUI
- Incorporating rhythm
- Chord recognition and generation

GUI Milestones

- Basic Frame
  - Where everything goes
- Existing Functions
  - Load Sample
  - Generate Music
  - Play Music
  - Save Music
- Extend to include settings
  - Change how music is generated
- Extend to include note visualizer
  - See generated notes
  - Modify generated notes
  - User interaction with music generation



Graphical User Interface
- **Basic Frame**
  - Generate mockup
  - Create main window
  - Create pane place holders
  - Create menu bar
  - Finished Basic Frame
- **Load/Save/Generate (Basic funct...**
  - Create button pane
  - Add load
  - Add generate
  - Add Save
  - Add Play
  - Add Exit to menu->file
  - Add Controller
  - Finished basic functions



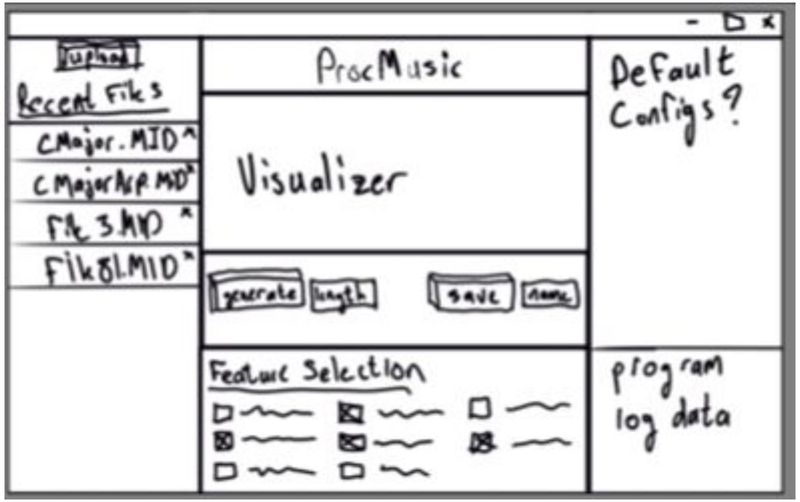
Settings/Modifiers
- Create Settings/Modifiers Pane
- Add settings components
- Modify Controller to send setting d...
- Finished Settings & Modifiers

Note Visualizer
- **Creation**
  - Create visualizer pane
  - Modify Controller to get note data
  - Show note data
- **Extra Functions**
  - Output regeneration
  - Output recycling
- **User Modificaiton**
  - Modifiy visualizer to allow for use...
  - Note additon
  - Note deletion
  - Recycle extension
- Finished Note Visualizer

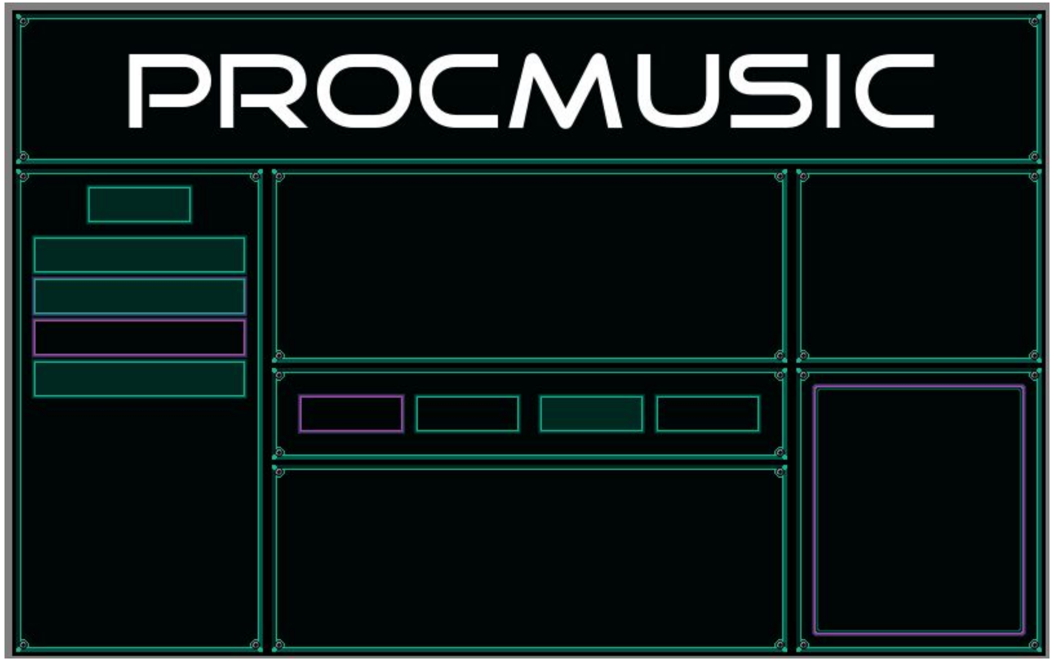
Finished GUI

Basic Frame : Mockups

Layout

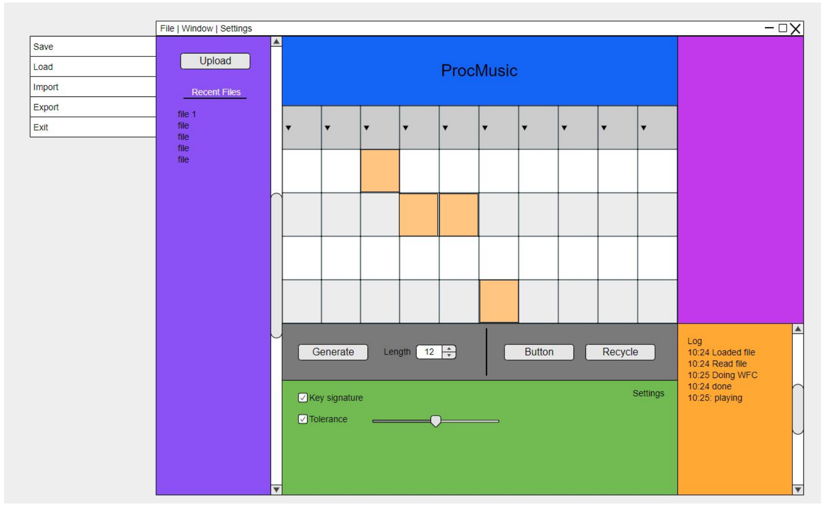


Visual

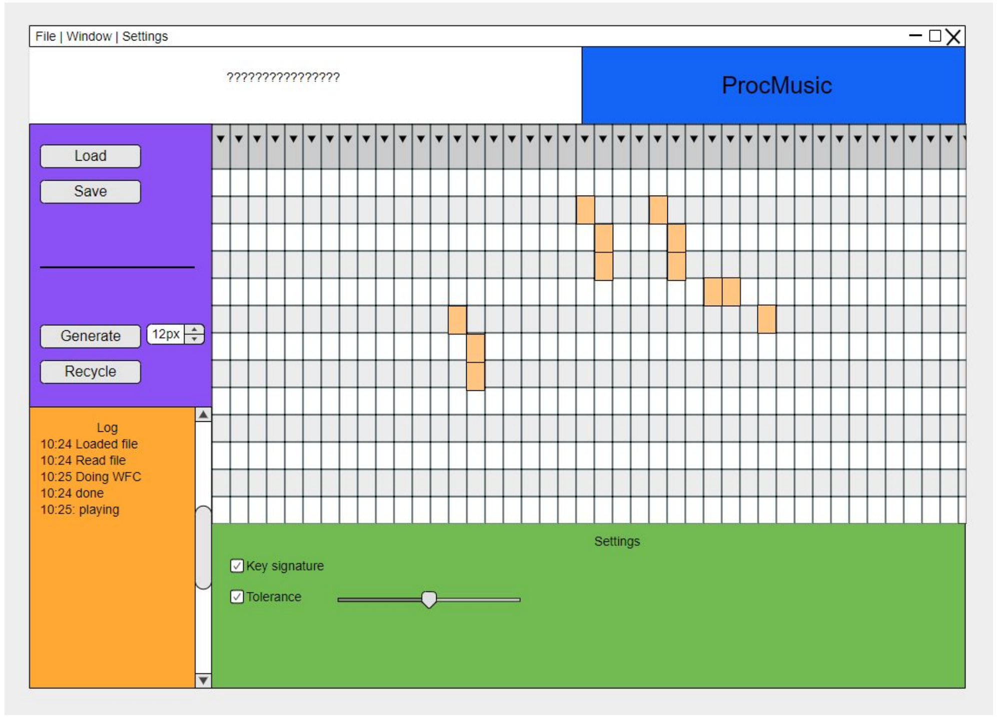


Default Hovered Clicked

Basic Frame : Mockups

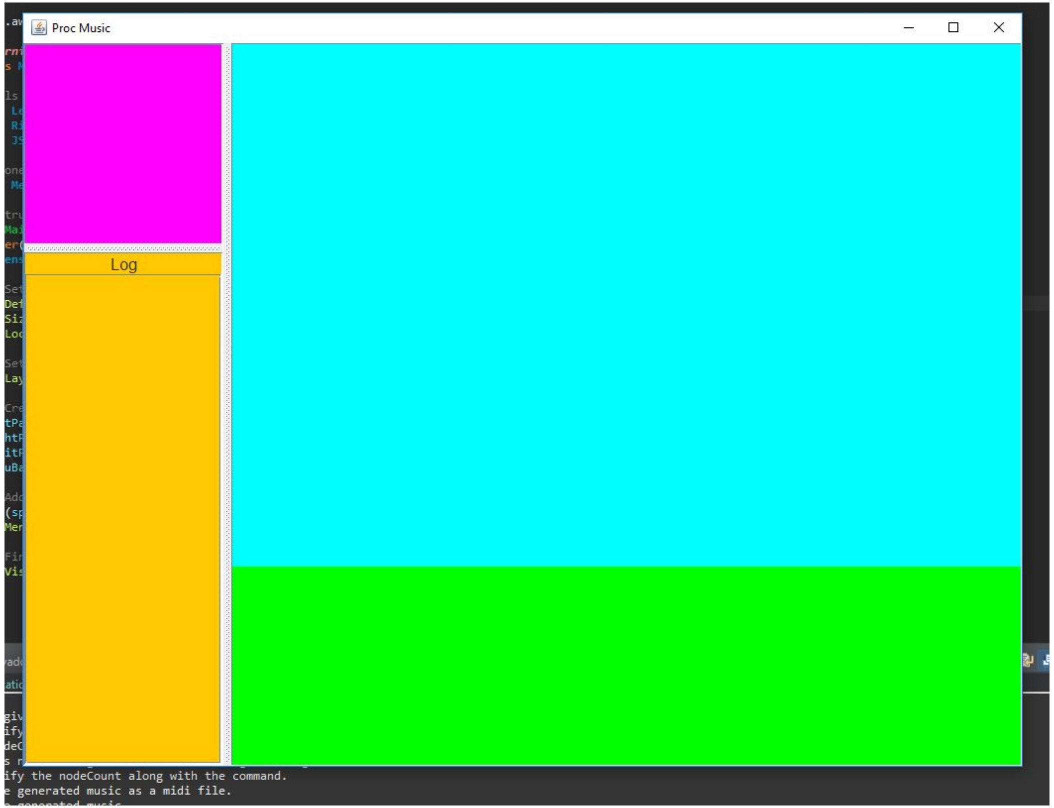


```
Log
10:24 Loaded file
10:24 Read file
10:25 Doing WFC
10:24 done
10:25: playing
```

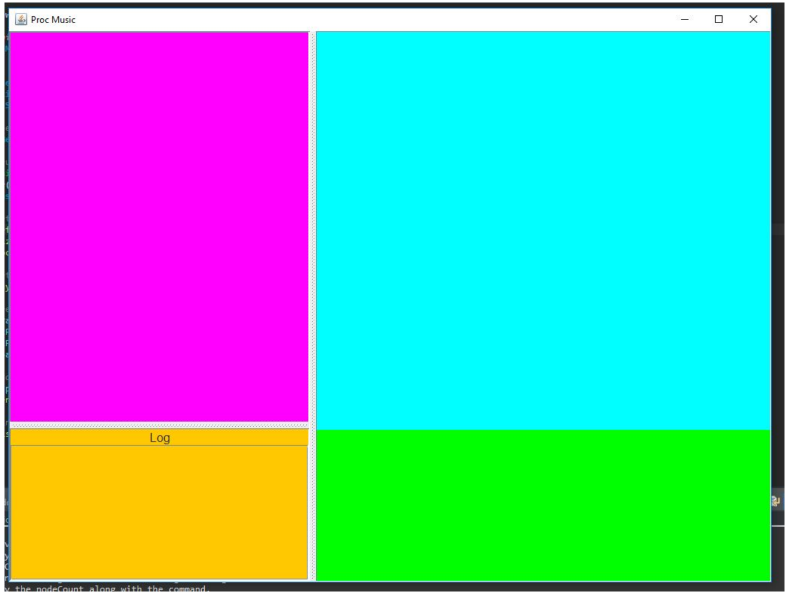


```
Log
10:24 Loaded file
10:24 Read file
10:25 Doing WFC
10:24 done
10:25: playing
```

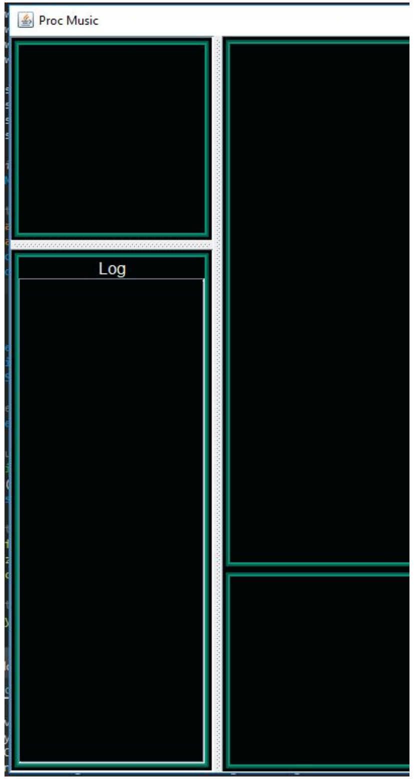
Basic Frame: Iterations

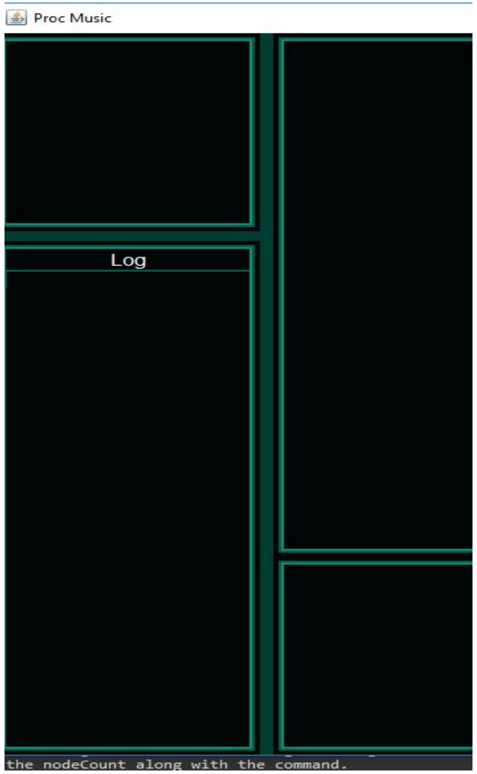


Panel size, layout and resizing behavior



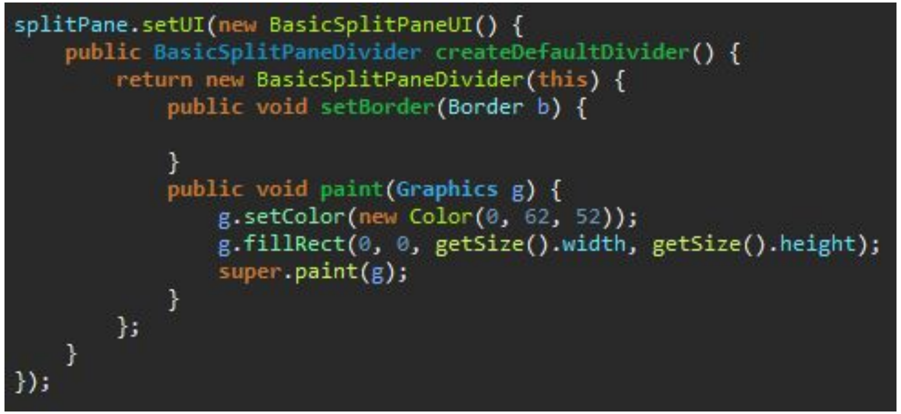
Basic Frame: Iterations





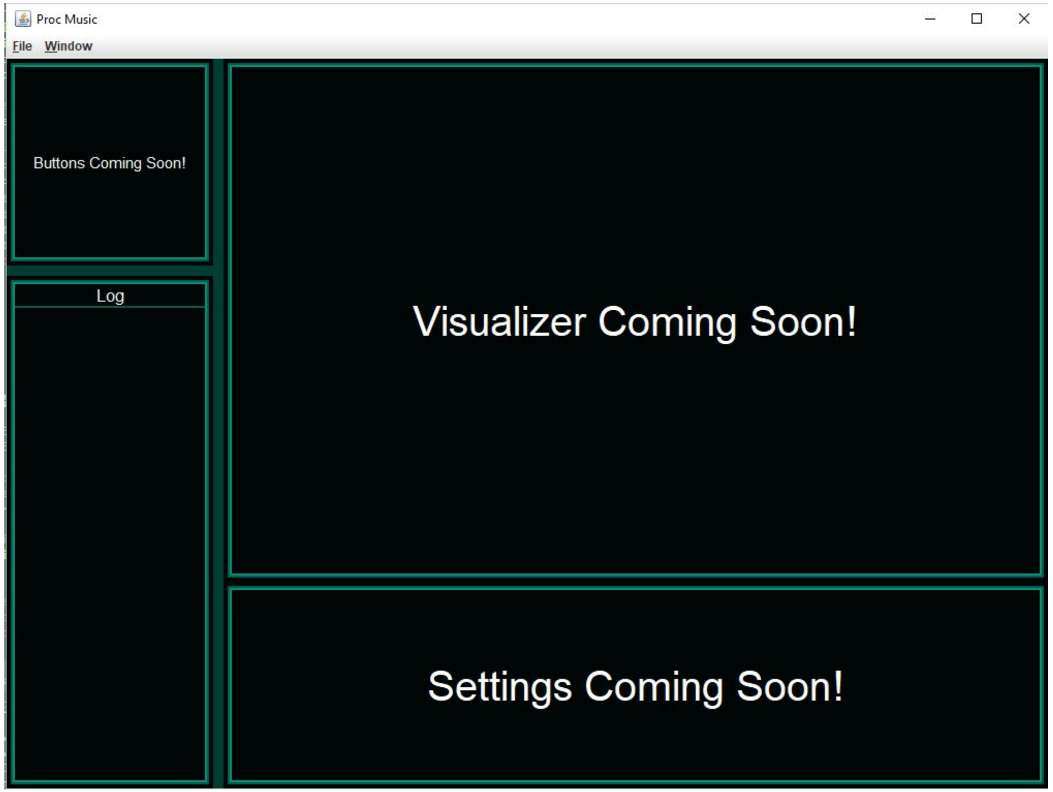
the nodeCount along with the command.

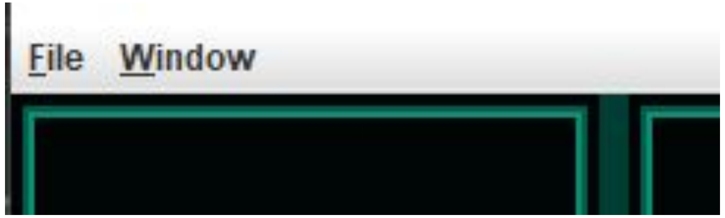
Visuals
Divider

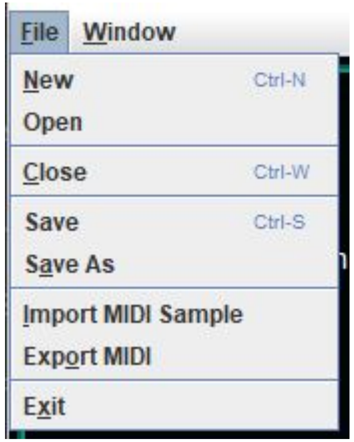


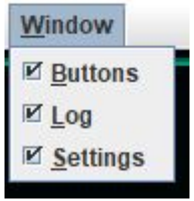
```
splitPane.setUI(new BasicSplitPaneUI() {
    public BasicSplitPaneDivider createDefaultDivider() {
        return new BasicSplitPaneDivider(this) {
            public void setBorder(Border b) {

            }
            public void paint(Graphics g) {
                g.setColor(new Color(0, 62, 52));
                g.fillRect(0, 0, getSize().width, getSize().height);
                super.paint(g);
            }
        };
    }
});
```

Basic Frame: Finished Product

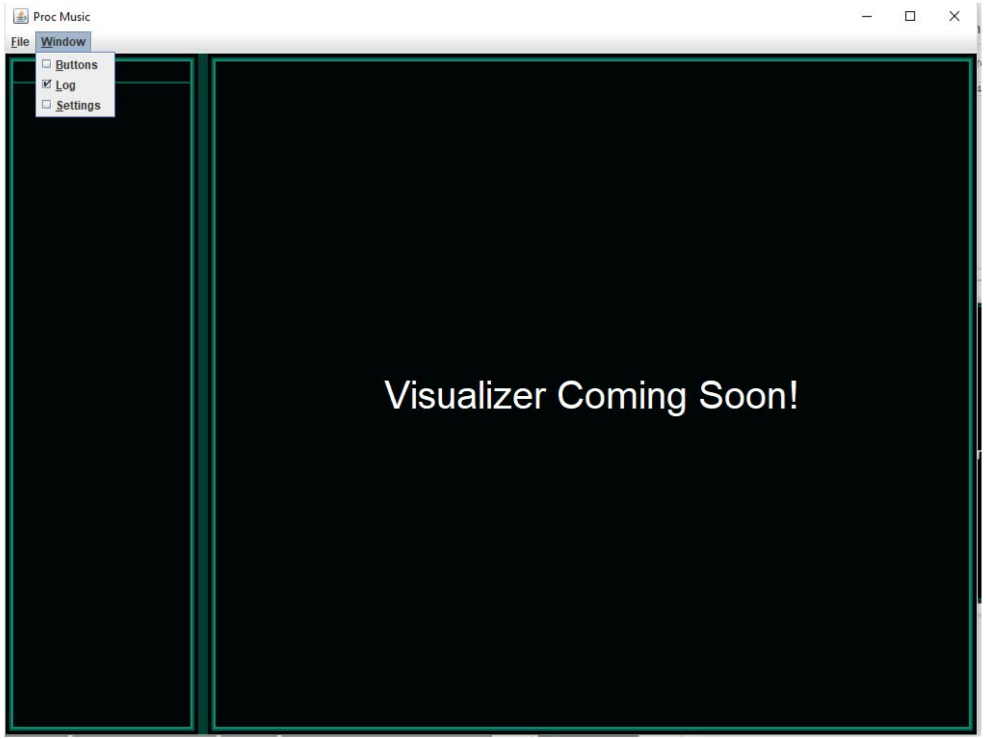


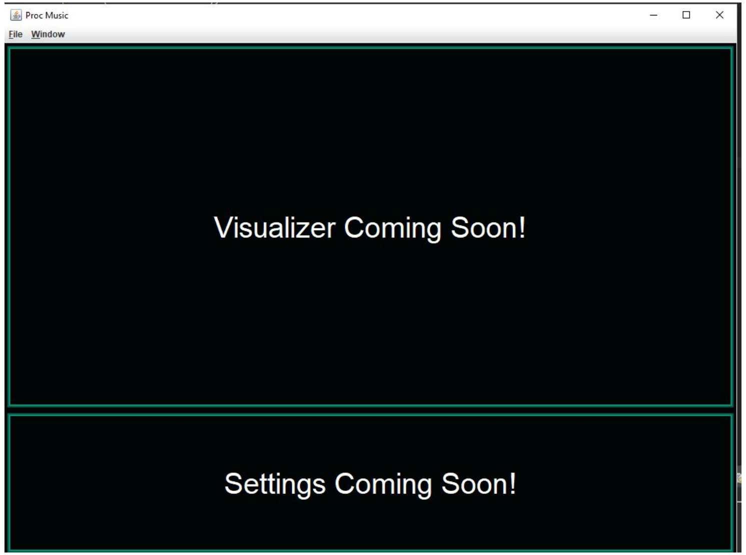




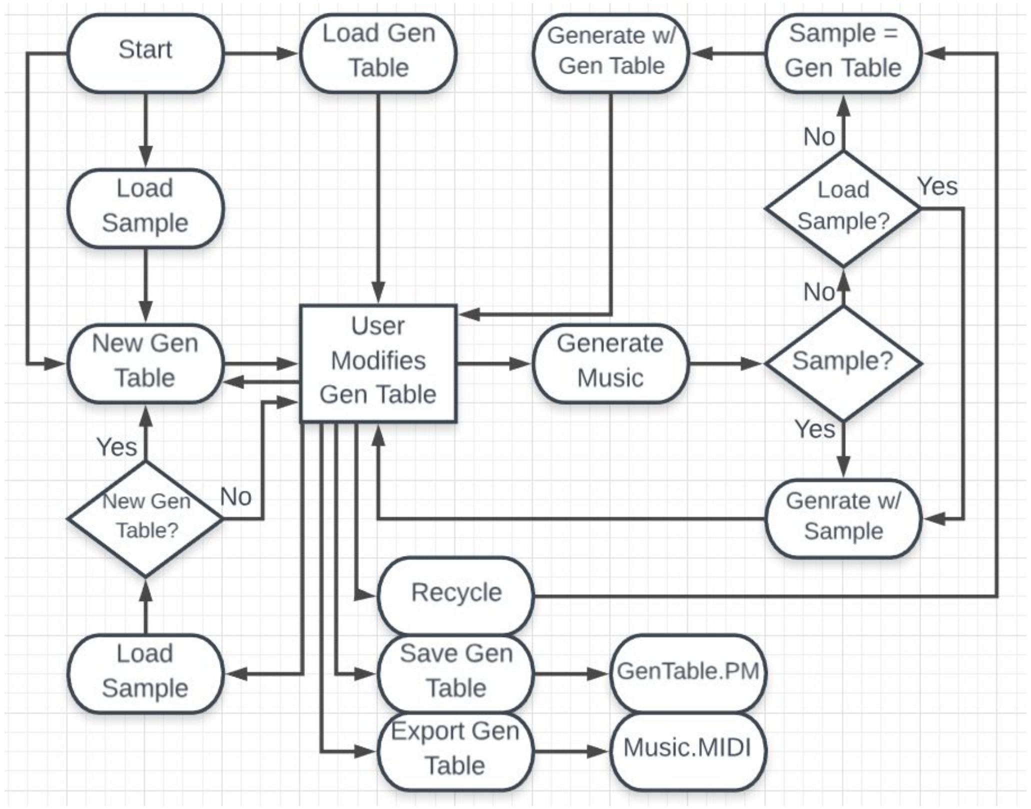


Basic Frame: Finished Product





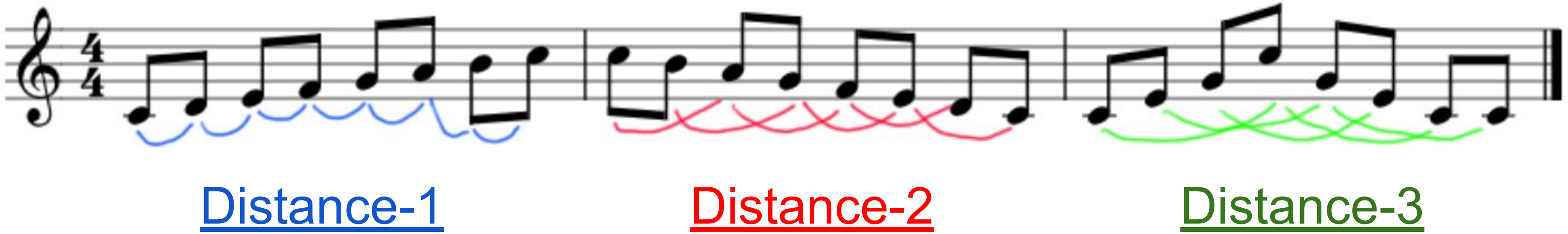
Action UML(ish)



- Basic frame done
- Prepare for existing and extra user actions and functions

Pitch Changes

- Extended distance-1 transitions to distance-n transitions
- Added ability to overlay multiple pitch modifiers

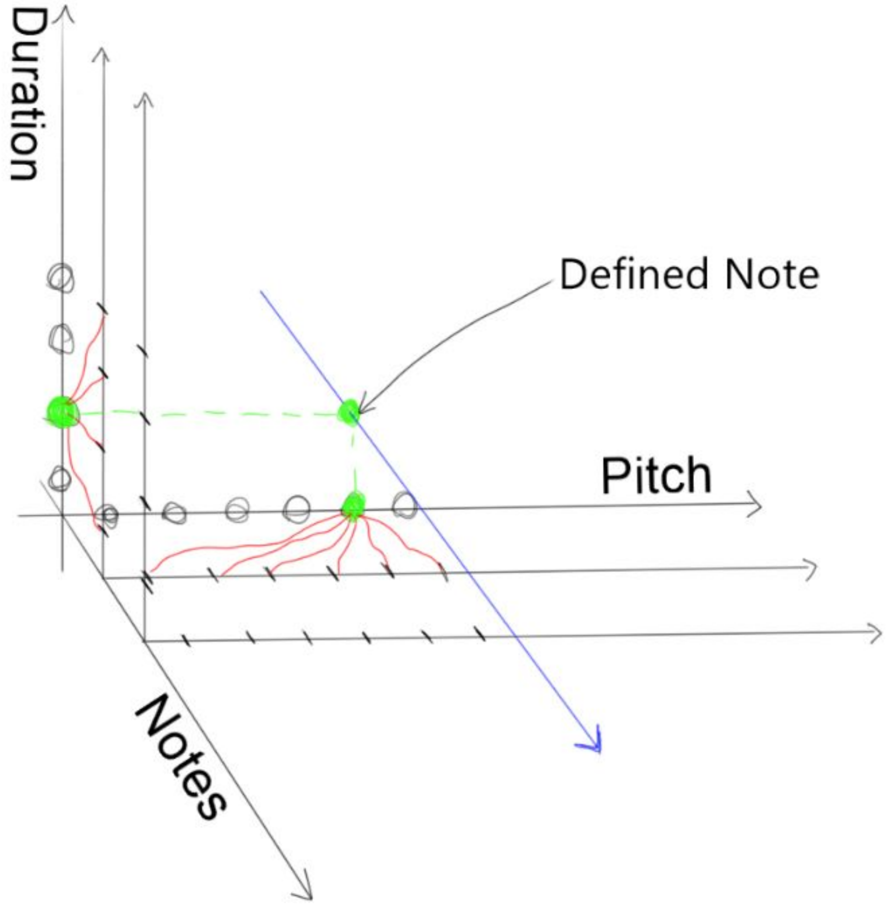


Note Duration

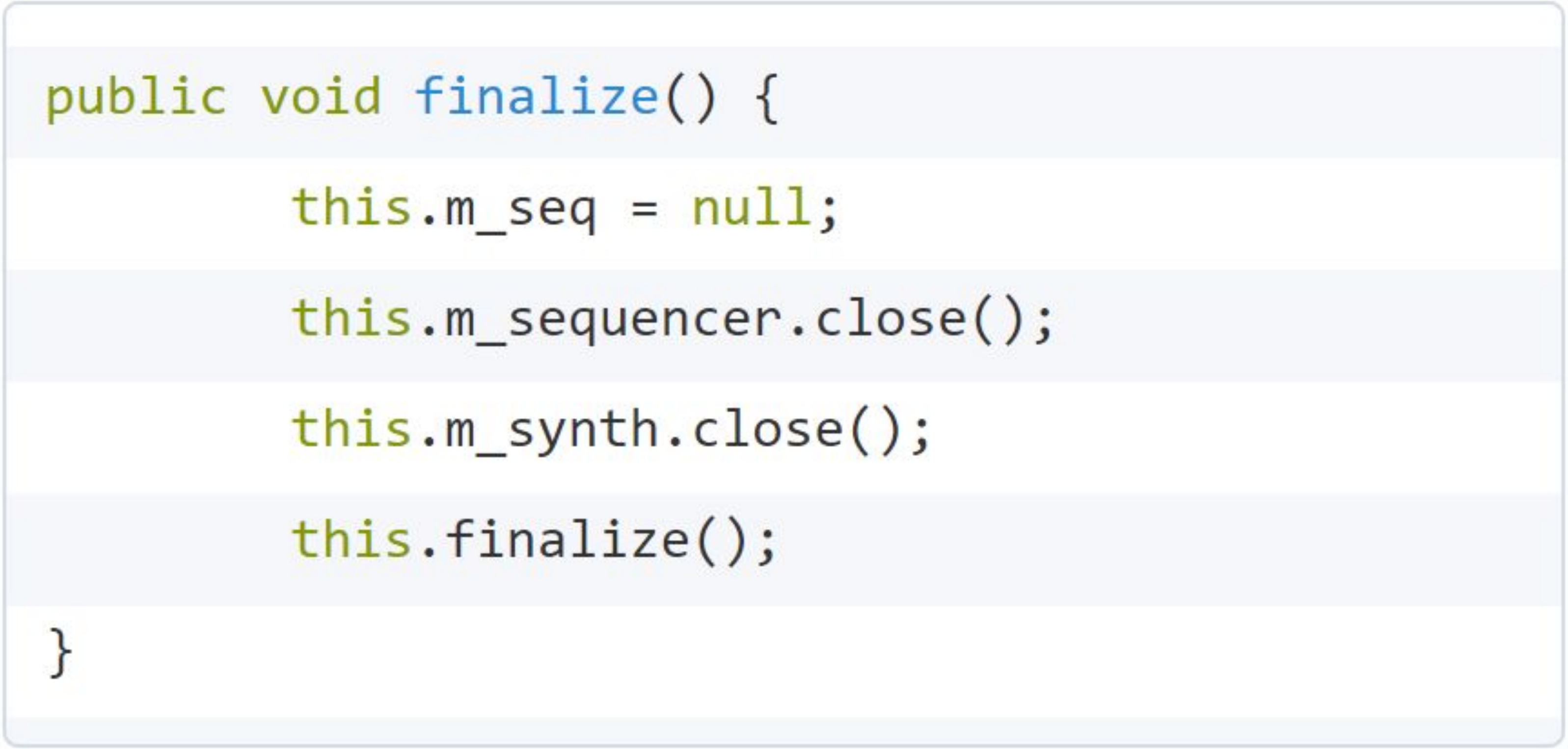
- Calculating rhythm probabilities on input samples
- Probabilistically generating notes with rhythms
  - No longer stuck with fixed-length notes
  - Uses wave function collapse

Wave Function Collapse

- Collapsing in higher dimensions
- Why WFC over simple Markov Chains
- Potential Problems



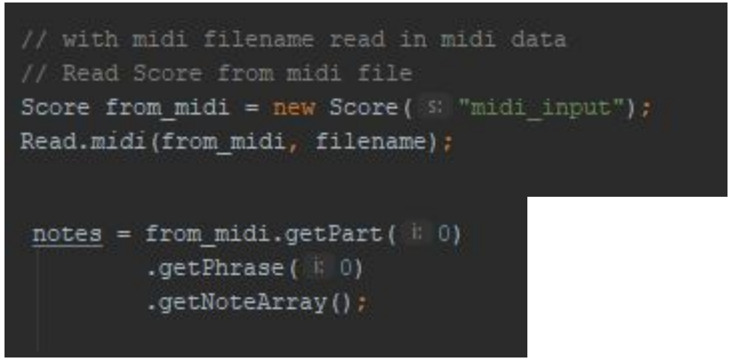
Fun Library stuff



```
public void finalize() {
        this.m_seq = null;
        this.m_sequencer.close();
        this.m_synth.close();
        this.finalize();
}
```

Integrating Chord Recognition and Generation

- Reading chords from MIDI files
  - Changing the way we read MIDI files
  - Was using jMusic API's built in methods, which had some shortcomings in terms of how it grouped note data from the MIDI file.

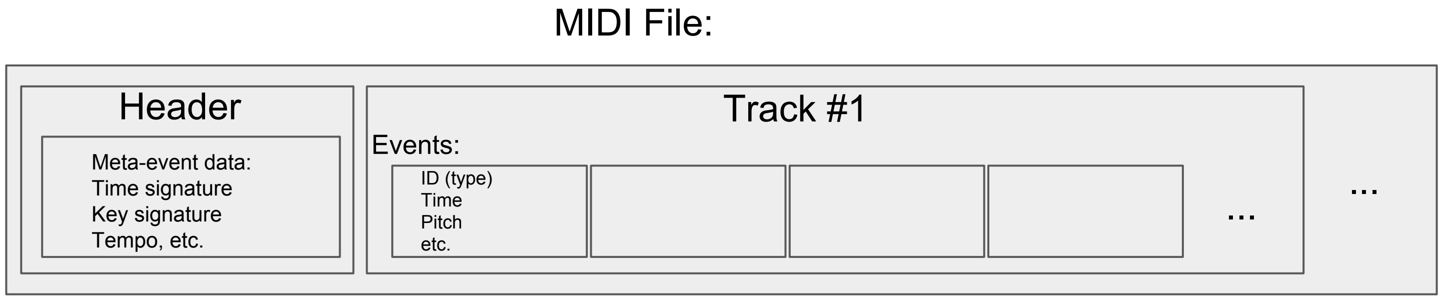


```
// with midi filename read in midi data
// Read Score from midi file
Score from_midi = new Score( s: "midi_input");
Read.midi(from_midi, filename);

 notes = from_midi.getPart( i: 0)
         .getPhrase( i: 0)
         .getNoteArray();
```

Issues:
Multiple phrases in a part whose notes are not given chronologically.
i.e., 1st note may be in 1st phrase, but the 2nd note in the 1st phrase might not actually be the 2nd note of the song.

MIDI events

- MIDI files made up of ‘chunks’. There are header chunks and track chunks.
- Track chunks have the actual music data. There is usually a track for each instrument in a MIDI file. Each tracks contain events, which hold the actual music data among other things. Luckily, jMusic provides a good way to sift through the raw MIDI data and get Event objects from the file.



Integration into music generation

- For determining chords, we are mostly concerned with NoteOn and NoteOff events.
- Probabilities generated and used similar to single Note pitches/durations. We’ll just be paying attention to arrays of pitch values rather than single pitches.

Next Steps

- Finish chords
- More GUI work
    - Support Existing functions
    - Settings
- Key signature detection
- User modification