Samuel Baldwin

Rah Hite

Griffin Miller

Giles Thomas

CSCI 205 Design Manual

Introduction: Using JMusic as our primary library, we've developed a keyboard that activates sound at different pitches, with the ability to to change note length, record a series of notes, and transpose octaves. Additionally, we've created a GUI which works with the code that allows a relatively comfortable user experience. Through the GUI, a user is able to change certain elements of the sound generated by the keyboard.

User Stories:

The user should be able to...

- Completed:
 - Produce synthesized pitches which are part of the 12-tone scale in real time
 - Tell whether a pitch is active
 - Control the volume of the outputted sound
 - The pitches are mapped to the keys on a UI keyboard
 - Record a composition
 - Reset the recording
 - While adjusting sliders, level numbers are displayed.
 - Control the length of a pitch
 - Reset the recording

- Partially Completed:

- Transpose
 - Explanation: The user can change octaves, but can't transpose by single pitches
- Interact with a sleek, simple, cohesive design for the keyboard & knobs
 - Explanation: UI design is complete, but knobs were never incorporated
- Tell what pitch they're playing through labels on keys (ex. C4, F#4, etc.)
 - Explanation: We couldn't find a way to label the keys, but we added a UI element that detects and reports which note the mouse is hovering over
- Record a composition
 - Explanation: We enabled the dictation of notes on an ASCII staff, but we weren't able to implement playback

- Not Completed:

- Have a choice over what sort of synthesizer waveform we can use (Triangle,
 Square, etc.)
- Control the timbre of the sound using a Cutoff knob
- Determine the speed of sound using an Attack knob
- Determine how quickly a pitch ends using a Release knob
- Cutoff, attack, and release can be controlled by UI elements
- Dropdown folder with choices of waveform shapes.
- Change the instrument sound produced when playing keys

Object-Oriented Design Overview

CRC Cards:

UIController

Responsibilities: To implement JMusic elements into an accessible user interface. Also, allow user to change how the output of the keyboard sounds

Collaborators: Sam Baldwin, Giles Thomas, Griffin Miller, Rah Hite

KeyboardMain

Responsibilities: To start the keyboard application.

Collaborators: Sam Baldwin, Griffin Miller

UML Diagram:

UIController extends Application

-btnOctaveDown:Button -btnOctaveUp:Button

-infNoteCheck:CheckBox

-sliderVolume:Slider -sliderNoteLength:Slider

-aKey:Rectangle -aSharpKey:Rectangle

-bKey:Rectangle -cKey:Rectangle

-cSharpKey:Rectangle

-dKey:Rectangle

-dSharpKey:Rectangle -eKey:Rectangle

-fKey:Rectangle

-fSharpKey:Rectangle

-gKey:Rectangle -gSharpKey:Rectangle

+ initialize() : void - recordingSwitch() : void

- initScore() : void

-updateScore(int note) : void -addPlayedNoteToScore(int note) : void

-addKeyPresses() : void

+addKeytracking(): void

-keyClickHandler(Rectangle rect, int noteValue, double noteLength) : void -addOctaves() : void

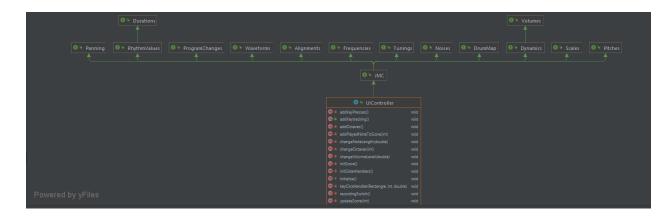
-changeOctaves(int octaveDiff) : void

-initSliderHandlers(): void -changeNoteLength(double newNoteLength): void -changeVolumeLevel(double newValue)

KeyboardMain

+ main(String[] args) : void + start(Stage primaryStage) : void

IntelliJ UML Diagrams:



Citations

Lab 11 instructions -

https://onedrive.live.com/?authkey=%21AFebZ5S9m6xl%2DFA&cid=8B4B8D135E3EEA4D&id=8B4B8D135E3EEA4D%2114391&parld=8B4B8D135E3EEA4D%211687&o=OneUp

¡Music tutorials -

https://explodingart.com/jmusic/jmtutorial/t1.html

jMusic documentation -

https://explodingart.com/jmusic/jmDocumentation/index.html

Synthesizer example -

https://www.youtube.com/watch?v=q09cNltGhLQ&list=PLuF1v-3Fw9goRMFFldza3RddAiwaZUqL5

Frame inside JavaFX (runs at negative frame rate, so had to be removed) - https://stackoverflow.com/questions/64616443/how-to-get-awt-window-for-javafx-stage