Phase 1 (Due Sunday 10/11):
Make the program listen for key presses for the appropriate amount of time given a bpm, noteInterval, and numMeasures. Output the recorded times of the key presses along with the expected times of the key presses.

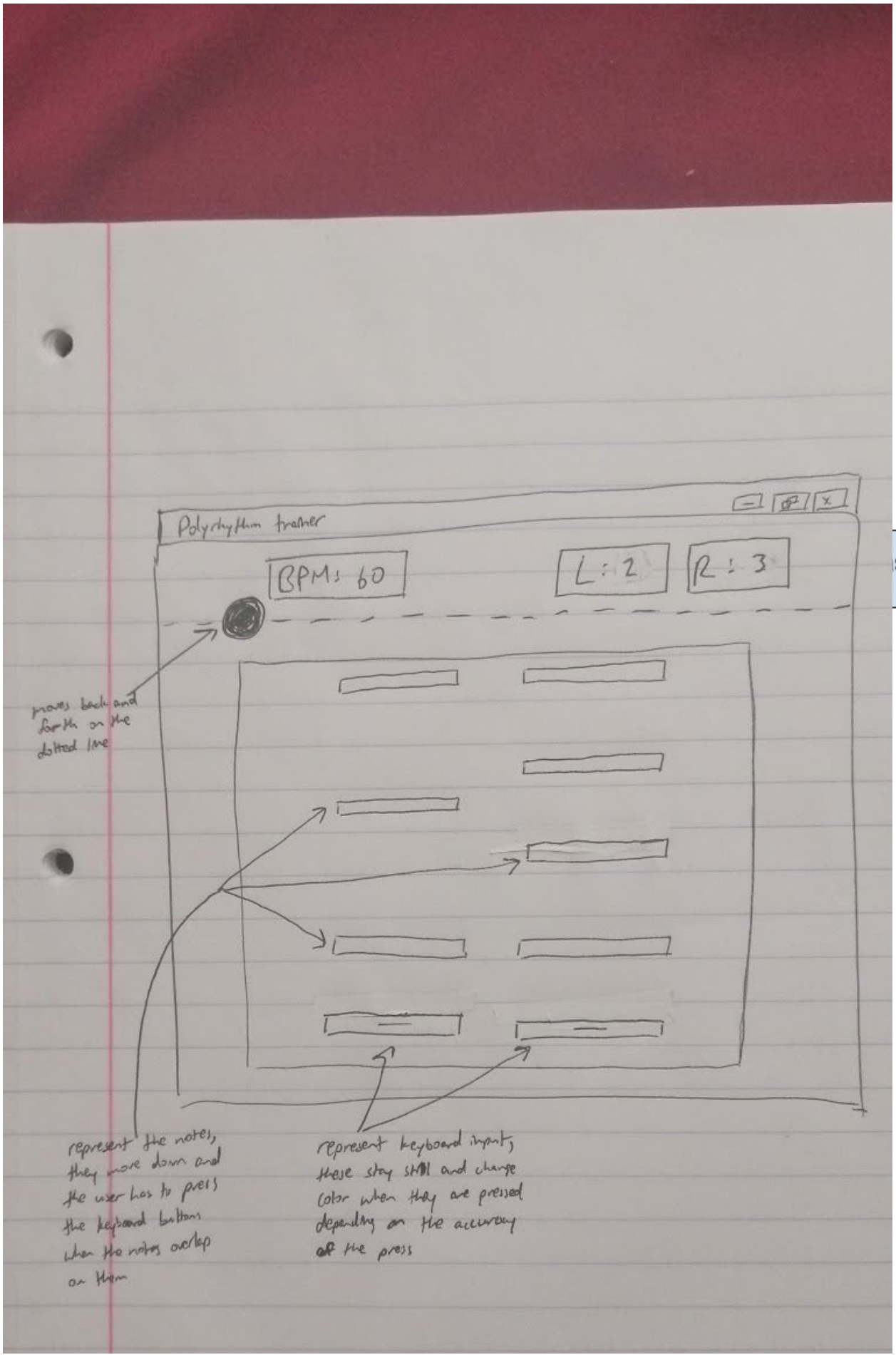
Phase 2 (Due Wednesday 10/14):
Implement the tempo ball and falling notes

Phase 3 (Due Sunday 10/18):
Finish product

Background info:
Let's take an example where you have a metronome running at 60 beats per minute. The music you listen to consists of mostly simple polyrhythms where one line of music has eighth notes and another line has sixteenth notes. Mathematically, this is very simple to play because the length of each eighth note is exactly equal to the length of two sixteenth notes. However, sometimes there will be two lines of music where one line is playing eighth notes and the other is playing triplets. Remember when you were learning about fractions in elementary school and you noticed that 1/2 + 1/4 is much easier to compute than 1/2 + 1/3? The same concept applies to music; it's much harder to understand when the triplets should be played relative to the eighth notes. This application is intended to train your muscle memory so that you will be able to play polyrhythms more comfortably.

List of features:
The user should be able to assign two buttons representing the rhythms of each line of music.

The application should contain a running metronome at the top who's BPM can be assigned to any value, and the progress of the metronome between each beat should be represented by a ball moving back and forth (the ball touches the end of the window when a new beat starts).


Polyrhythm trainer
BPM: 60
L: 2
R: 3
moves back and forth on the dotted line
represent the notes, they move down and the user has to press the keyboard buttons when the notes overlap on them
represent keyboard inputs, these stay still and change color when they are pressed depending on the accuracy of the press

sures: 8

High level code design:

Static Class Settings that store the bpm, leftKey, leftInterval, rightKey, rightInterval, and numMeasures

Class WindowMaker that displays everything. All the supporting classes are called from this class.

Class InputAnalyzer that tells WindowMaker and ResultAnalyzer how accurate the key input was to the beat

Class ResultAnalyzer that gives WindowMaker information on the user's performance after the Action is over

Low level code design:

"The Sequence" refers to the events that start when any key is pressed: the tempo ball starts moving and the falling notes appear at the top of the screen. The user is given 4 beats of time before the falling notes reach their button and they have to input their first key

"The Action" refers to the events that start when the user inputs their first key

<u>class WindowMaker</u>

```
Function that listens for key input to start the Sequence
{
        sf::Clock sequenceClock
        // sfml clock that acts as a timer
        // tracks the elapsed time since any key was pressed

        Listen for any key input
        When a key is pressed, call
}

Function that displays tempo ball
{

}

Function that displays moving notes
{

}
```

```
Function that listens for key input when the Action begins
{
        sf::Clock actionClock
        // sfml clock that acts as a timer
        // tracks the elapsed time since the Action began

        int sequenceDuration
        // how long the sequence is supposed to be
        // provides a stopping point for the while loop that listens to key input

        Listen for key input while clock's elapsed time < sequenceDuration
                When a key is pressed, check if it is the escape key
                If escape:
                        tell InputAnalyzer to reset the info in ResultAnalyzer
                        call function that resets display to starting position
                If not escape:
                        call InputAnalyzer function and pass the key and elapsed time to the
                function:
                                If the function returns "LeftRed" then color the left button red
                                If the function returns "LeftYellow" then color the left button yellow
                                etc...
}


class InputAnalyzer

int bpm

[string] leftKey
int leftInterval
// notes per beat on the left hand
// e.g. leftInterval = 3 means the left hand is playing triplets

[string] rightKey
int rightInterval

ResultAnalyzer results
//

Function that takes a key and elapsed time and returns how accurate the input was in the form
of "leftRed / leftYellow / leftGreen / rightRed / rightYellow / rightGreen"
{
        Example: bpm = 120, leftInterval = 2, rightInterval = 3
```

The first accurate input for the left hand would be at (60 / bpm) * (1 / leftInterval)
}

Function that returns ResultAnalyzer to WindowMaker so WindowMaker can display the results
{
return results
}

Function that resets the info in ResultAnalyzer
{

}

class ResultAnalyzer