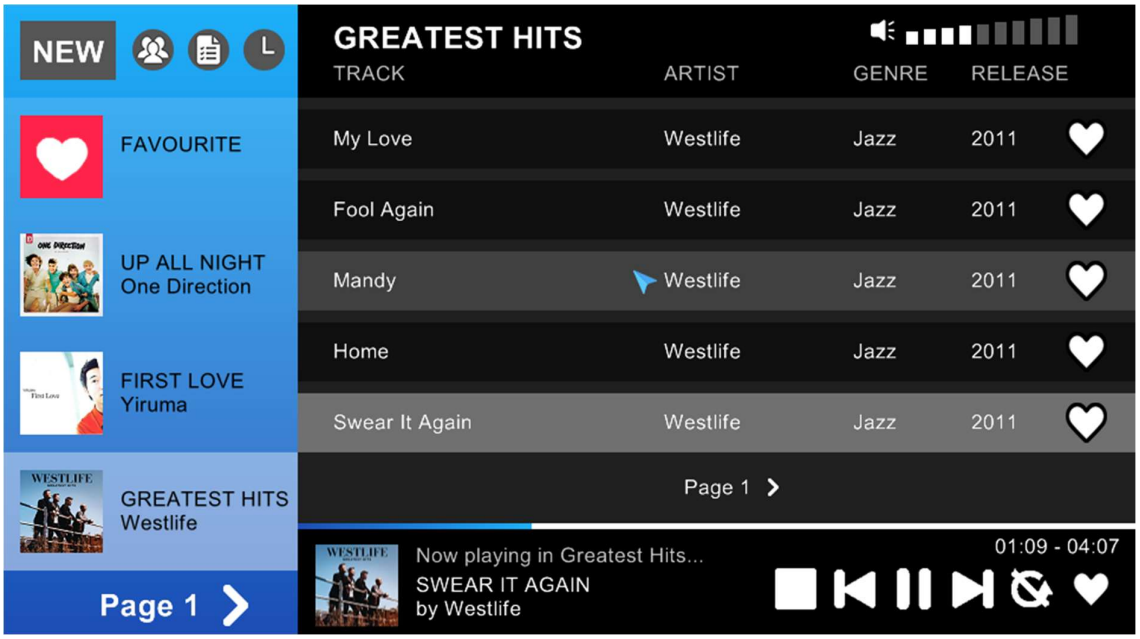
Custom Project

Name: Nguyen Thi Thanh Minh
Student ID: 104169617

Song Progress Bar Drawing in Gosu with AudioInfo Gem Tutorial

Introduction

When building a graphical music player, it's important to provide visual feedback to the user about the playback progress of the currently playing song. A progress bar is a simple yet effective way to display this information in the interface. By visualizing the elapsed and remaining time, the progress bar allows the user to clearly see how much of the song has played already, and how much is left.



A playing song with progress bar and elapsed time – total length information shown on GUI

In this tutorial, we will learn how to dynamically draw a progress bar that reflects the playback position of the song. To accomplish this in Ruby, we will use the Gosu library for building the GUI, along with the *ruby-audioinfo* gem for getting audio metadata like length. Gosu provides the graphics capabilities to render the progress bar, while *ruby-audioinfo* allows us to easily get the total song length in seconds (together with many other characteristics of the song).

The steps will include:

1. Installing dependencies
2. Loading the audio file and getting the song length
3. Calculating elapsed time

4. Drawing the progress bar
5. Displaying in digits

Step 1: Installing and importing dependencies (*ruby-audioinfo* gem)

ruby-audioinfo is a Ruby gem that glues together various audio ruby libraries and presents a unified API to the developer. Currently, the supported formats are: mp3, ogg, mpc, ape, wma, flac, aac, mp4, m4a. It does not depend on any gems with native extensions for portability.

To install the gem, simply run this command in terminal:

```
gem install ruby-audioinfo
```

Then require the gem in the .rb file at the start of the code:

```
require 'audioinfo'
```

This allows the user to access to many information of an audio file, such as artist, title, bitrate, as well as length. In this tutorial, we will focus on the audio length only.

Step 2: Loading an audio file and getting its length

We can now initialize a variable in the class *AudioInfo* and load an audio file from some directory path, i.e. `info = AudioInfo.open("path/file.supported_extension")`

For example, in my GUI Music Player program, I keep my song files (mp3) in different album folders, then store their specific location in the attribute *:location*. Therefore, when playing a song, I first assign an *info* variabe:

```
@playing_track = @selected_track
info = AudioInfo.open(@playing_track.location)
```

And get the song's total time length in seconds with the *length* method:

```
@playing_track_length = info.length
```

Now we've got the necessary information. The next question is how to make use of this to visualise the current song progress.

Step 3: Calculating elapsed time

Before thinking about how to actually draw a progress bar, we need to calculate the elapsed time of the playing song first. In other words, we need to know how long has the song been played (excluding when being paused). For this, we can use a method from *gosu*:

```
@time = Gosu.milliseconds
```

This returns the number of milliseconds elapsed (integer). We calls this method whenever playing a song to initialize a starting time. Then, we constantly compare this with the current time (also by calling **`Gosu.milliseconds`**) when the song is playing. The value is then converted to seconds through the division by 1000, and the result we have now is elapsed time.

To express it in maths: elapsed time (secs) = (current time (millisecs) - starting time (millisecs)) / 1000

In code, for example:

```
@count_up = (Gosu.milliseconds - @time)
seconds = @count_up / 1000
```

When the song is being paused , we simple refresh the starting time in the way that keeps the distance with the initial starting point:

```
@time = Gosu.milliseconds - @count_up
```

Now, we can use these values to calculate the instant width of the progress bar.

Step 4: Calculating width and drawing the progress bar

In this calculation, we will use these values: unchanged audio total time (in seconds) and maximum bar width, and constantly changing elapsed time (also in seconds).

Expression: current width = max width * elapsed time / audio total time

```
@progress = seconds * (PANEL_WIDTH) / @playing_track_length
```

Complete function:

```
def create_progress_bar()
    if @music_player.playing?
        @count_up = (Gosu.milliseconds - @time)
        seconds = @count_up / 1000
        @progress = seconds * (PANEL_WIDTH) / @playing_track_length
    else
        @time = Gosu.milliseconds - @count_up
    end
end
```

After that, draw a rectangle (in *draw* function) with found width above a blank progress bar of full width. In the belowing example, I use the *draw_quad* method for color effect:

```
draw_quad(x, y, BOTTOM_COLOR, x, (y+5), BOTTOM_COLOR, (x+@progress), (y+5), TOP_COLOR,
(x+@progress), y, TOP_COLOR, ZOrder::UI, mode=:default)
```

And the result is like the screenshot at the beginning.

Step 5: Also displaying it in mm:ss format onto the screen

Another helpful visualization of the song progress (also shown in the screenshot) is the number display of elapsed time – total time in mm:ss format.

First, we need to convert the elapsed time from float into integer.

Second, divide it by 60 (which returns only the whole part) to get the minutes value, and format it into a two-digit string with **`'%02d' % <integer>`**

Next, modulo-divide the elapsed time by 60 (which returns only the remainder) to get the seconds value, and format it.

Do the similar things with the total time, then use *draw_text* to put them onto the screen.

```
progress = (@count_up / 1000).to_i
progress_min = '%02d' % (progress / 60)
progress_sec = '%02d' % (progress % 60)
total_min = '%02d' % (@playing_track_length / 60)
total_sec = '%02d' % (@playing_track_length % 60)
MEDIUM_TEXT.draw_text("#{progress_min}:#{progress_sec} - #{total_min}:#{total_sec}",
WIDTH - 120, y+10, ZOrder::UI, 1.0, 1.0, WHITE)
```

Conclusion

Now you understand how to visualize song playback to create an intuitive music player interface in Ruby. The progress bar and digit display is a simple yet effective feature that improves the user experience of a music player.

Reference

https://github.com/moumar/ruby-audioinfo#readme

https://www.rubydoc.info/gems/gosu/Gosu.milliseconds