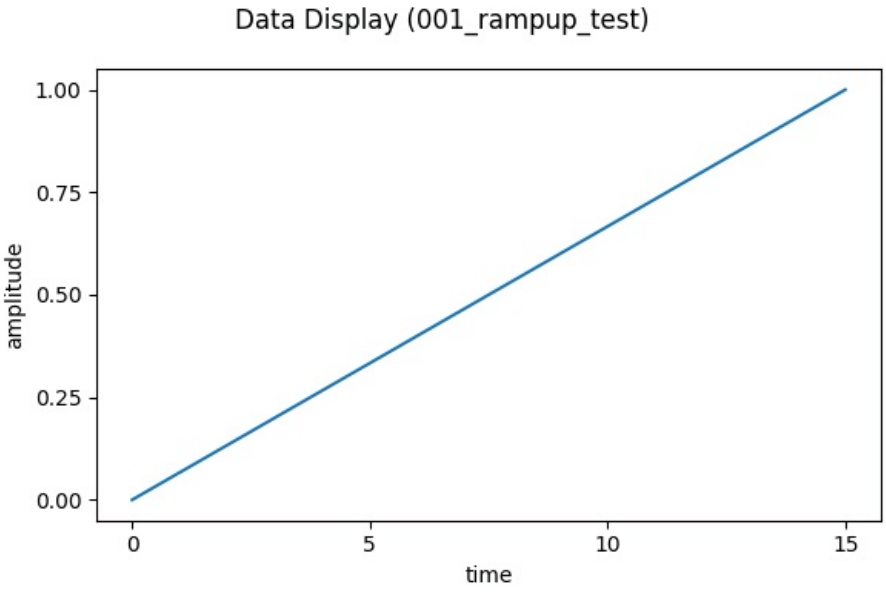
CS 3005: Programming in C++

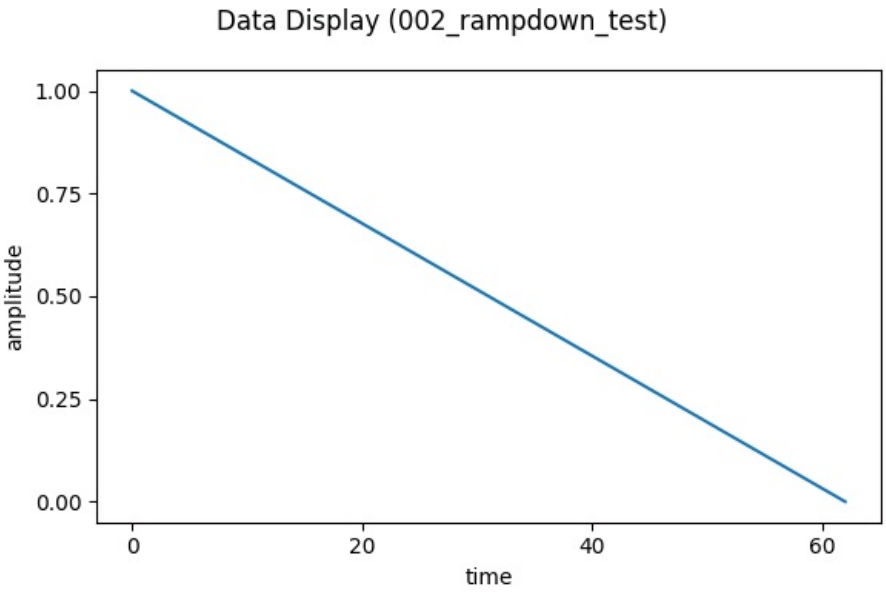
Audio Track Creator

Introduction

Audio tracks can be configured by sample rate (samples per second) and by duration (number of seconds). After configuration, audio tracks can be filled with audio data. For audio data to be converted to interesting sounds, there are usually patterns to the values stored.

One simple pattern for values is a “ramp”. A ramp smoothly changes its values from the beginning to the end. Below are examples of ramp up and ramp down.





The mathematical definition of a ramp is:

Given: `v1` and `v2` at the beginning and ending values in the ramp, `s` is the number of entries in the ramp, and `i` is the current entry in the ramp;

Then: `v(i) = v1 + (v2 - v1) * i / (s - 1)`

Assignment

In this assignment, you will make a program that allows the user to configure an audio track, and fill it with

either a ramp up or a ramp down pattern. The program will ask the user for samples per second and number of seconds. It will then ask the user whether the audio track should be filled with ramp up or ramp down.

After applying the configuration and inserted the ramp data, the program should display the contents of the audio track. The format of the display will look like this:

```
sample_number,amplitude
0,0
1,0.0666667
2,0.133333
3,0.2
4,0.266667
5,0.333333
6,0.4
7,0.466667
8,0.533333
9,0.6
10,0.666667
11,0.733333
12,0.8
13,0.866667
14,0.933333
15,1
```

Note there is a header line with `sample_number,amplitude`, followed by one line per entry. The commas in the output cause the data to be comma separated, and appropriate for creating CSV files. You can put this output into a CSV file, by copying and pasting it.

Visualize your output using the `show_graph.py` program we have provided.

```
./show_graph.py --data-file file_name.csv
```

Example Session

```
$ ./program-audio-track-creator/audio_track_creator
Samples/Second: 13
Seconds: 1.3
Fill style: rampup

sample_number,amplitude
0,0
1,0.0666667
2,0.133333
3,0.2
4,0.266667
5,0.333333
6,0.4
7,0.466667
8,0.533333
9,0.6
10,0.666667
11,0.733333
12,0.8
13,0.866667
14,0.933333
15,1
```

Programming Requirements

Update `library-application/ApplicationData.{h,cpp}`

`ApplicationData` Class

Data Members:

- `AudioTrack` Audio track associated with the current application. Should be created with the default constructor inside `ApplicationData`'s constructor.

`public` Methods:

- `AudioTrack& getAudioTrack();` Returns a reference to the audio track member.
- `const AudioTrack& getAudioTrack() const;` Returns a const reference to the audio track member.

Since `ApplicationData` uses `AudioTrack`, which library should be built first?

Create `library-commands/audio_track_creator_aux.{h,cpp}`

Functions:

- `void rampup_fill_audio_track(ApplicationData& app_data);` Fills the audio track of the given `ApplicationData` with smooth, linearly increasing samples. The samples should start at `0.0` and end at `1.0`. The number of entries in the ramp is the size of the vector.
- `void rampdown_fill_audio_track(ApplicationData& app_data);` Fills the audio track of the given `ApplicationData` with smooth, linearly decreasing samples. The samples should start at `1.0` and end at `0.0`. The number of entries in the ramp is the size of the vector.
- `void display_audio_track(ApplicationData& app_data);` Displays all of the samples of the audio track. Each sample is displayed in the following format: `sample_number,amplitude\n` where `sample_number` is which sample is being shown, `amplitude` is the value of the sample, and `\n` represents a newline. *NOTE:* They should be preceded by a header with a blank line, followed by a line indicating the format: i.e. the string `\nsample_number,amplitude\n`.
- `void fill_audio_track(ApplicationData& app_data);` Asks the user "Fill style: ". Depending on the string typed by the user, fills with rampup, fills with rampdown, or displays the message "Fill style 'bad-choice' is not allowed.".
- `int audio_track_creator(ApplicationData& app_data);` Asks the user "Samples/Second: " and "Seconds: ". The first is an integer, the second is a double. If both are positive, then sets the size of the audio track, fills the audio track, and displays the audio track. Otherwise, displays the message "Positive values expected for samples per second and seconds." Returns the size of the audio track.

Update `library-commands/Makefile`

Add `audio_track_creator_aux.{h,cpp}` in the appropriate places to add them to the library and install the header file.

Create `program-audio-track-creator/audio_track_creator.cpp`

Functions:

- `int main();` Entry point to the audio track creator program. Should create an `ApplicationData` and pass it to the `audio_track_creator` function found in `audio_track_creator_aux` and return the result of that function call.

Create `program-audio-track-creator/Makefile`

This file must contain rules such that any of the following commands will build the `audio_track_creator` program:

- `make`
- `make all`
- `make audio_track_creator`

Create `program-audio-track-creator/.gitignore`

The file `program-audio-track-creator/.gitignore` needs to store one line of text:

```
audio_track_creator
```

This will prevent the executable program `audio_track_creator` from being committed to the repository. It is a *derived file*.

Update `Makefile`

- Update the project-level Makefile so that `make` and `make all` in the project directory will call `make` in the `program-audio-track-creator` directory.

- If necessary, make sure the order of make commands is correct to build prerequisite libraries in the correct order.

Additional Documentation

- Function Overloading

Grading Instructions

To receive credit for this assignment:

- your code must be pushed to your repository for this class on GitHub
- all unit tests must pass
- all acceptance tests must pass
- all programs must build, run, and execute as described in the assignment descriptions.

Extra Challenges (Not Required)

- Create a `random_fill_audio_track` function. This function fills the audio track with random samples. Make sure `audio_track_creator` can call this function. Is this the same as white noise?
  - Random Double in C++ in range of 0-1
  - White noise
- Create an `exponential_rampdown_fill_audio_track` function. This function works similarly to the `rampdown` one, but decreases exponentially (i.e. divides by 2 each step). Try to make it still start at `1.0`.
  - Can you make the `exponential_rampup_fill_audio_track` function? If you want the function to end at `1.0`, can you calculate the value to start at? Does it work for small audio track sizes? Does it work for large ones? Does it work for very large ones? Why or why not?
  - Numeric limits of different C++ types
  - Double-precision floating-point format
  - Visualizer for what the bits looks like in a double