CS 3005: Programming in C++

Audio Track Creator (Additional Features)

Introduction

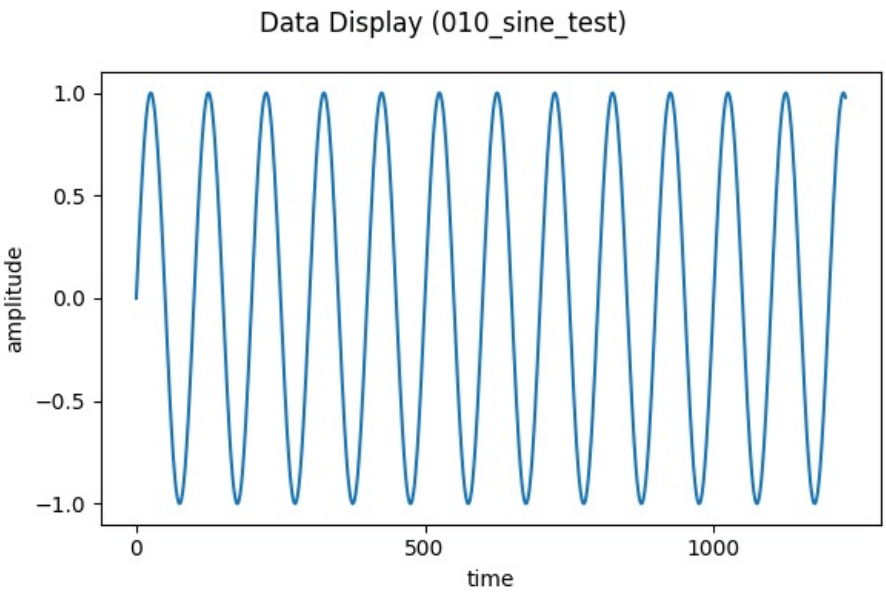
In the previous assignment, you created a program that creates an `AudioTrack` object and fills it with data, depending on the user's choice. In this assignment, you will add additional choices for the form of data that can be added to an `AudioTrack` object.

The two options are sine and sawtooth waves. The sine wave is generated using the `std::sin(angle)` function from the C++ standard library. An angle must be computed that corresponds to a position on the horizontal axis of the graph.

Given `sample_number` (the position in the audio track), `frequency` (how quickly the wave changes), and `samples_per_second` (the sample rate)

Then: `angle = (6.28 * sample_number * frequency) / samples_per_second`

The following is a graph for `samples_per_second = 1000` over a `1.23` second interval, at `frequency = 10.0`.



A sawtooth wave also follows a periodic pattern. But, it has straight line edges, that produce a pattern that resembles the teeth of a saw. The

Given: `sample_number` (the position in the audio track, an `unsigned int`), `frequency` (how quickly the wave changes, a `double`), and `samples_per_second` (the sample rate, an `int`)

Then:

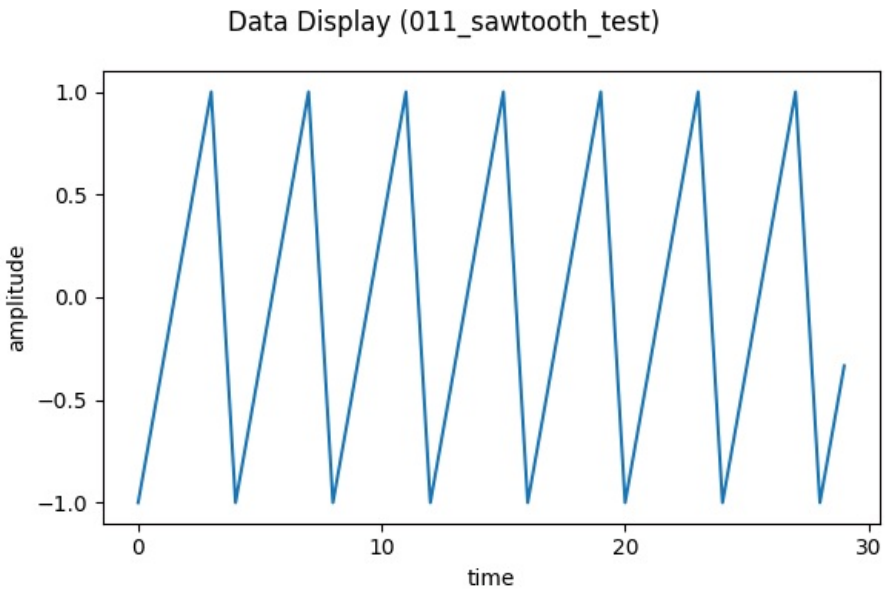
- `cycle_size = samples_per_second/frequency` is how many samples there are per cycle.
- `j = sample_number % cycle_size` is the how far into the current cycle the current sample is.
- `amplitude = -1.0 + (2.0 * j) / (cycle_size - 1)` is the amplitude of the sawtooth wave.

Even though `frequency` is a `double`, we want `cycle_size` to be an `unsigned int`. So, we assign the result of the division into `cycle_size` and intentionally discard any fractional part of the result. This is necessary because we want a discrete number of samples per cycle.

Now, when we compute `j` as an `unsigned int`, the `%` operator will work find with two integer types as operands.

Finally, `amplitude` is a `double`, that is computed from some float and some integer inputs. When `(2.0 * j)` is computed, `j`'s value is promoted to `double` to match the type of `2.0`. When the division `(2.0 * j) / (cycle_size - 1)` happens, the numerator is a `double` because it's the result of `2.0 * j`. The denominator's value will be promoted to `double` to match the type of the numerator. So, the overall computation results in a correct `double` value.

The following is a graph for `samples_per_second = 23` over a `1.34` second interval, at `frequency = 5.0`.



Assignment

In this assignment, you will update the `audio_track_creator` program to allow the user to select sine or sawtooth wave data in addition to the ramp options that were made available in the previous assignment.

An example interaction with the program could look like this:

```
Samples/Second: 1000
Seconds: 1.23
Fill style: sine
Frequency: 10

sample_number,amplitude
0,0
1,0.0627587
2,0.12527
3,0.187287
...
1228,0.988864
1229,0.977574
```

Another example interaction with the program could look like this:

```
Samples/Second: 1000
Seconds: 1.23
Fill style: whammy
Fill style 'whammy' is not allowed.
```

Programming Requirements

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

`ApplicationData` Class

Data Members:

- `std::vector<double> doubleRegisters;` Vector of 'registers' to store values for the application. Should be initialized in the constructor to have 5 register slots (size of 5).

`public` Methods:

- `double getDoubleRegister(const unsigned int position) const;` Returns the value of the given register. If `position` is out of range, return -INFINITY.

- `void setDoubleRegister(const unsigned int position, const double value);` Assigns the value of the given register to the given value. if `position` is out of range, do nothing.

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

Functions:

- `void sine_fill_audio_track(ApplicationData& app_data);` Fills the audio track of the given `ApplicationData` with a sine wave. The formula to get the amplitude at each sample should be as follows: `std::sin(angle)`. `angle` is computed using the formula described above. `samples_per_second` is the value previously assigned to the audio track and `frequency` is the value stored in the `ApplicationData`'s register 0.
- `void sawtooth_fill_audio_track(ApplicationData& app_data);` Fills the audio track of the given `ApplicationData` with a sawtooth waveform. The formula to get the amplitude of the sawtooth wave is described above. The `samples_per_second` and `frequency` are found as in `sine_fill_audio_track`.
- `void fill_audio_track(ApplicationData& app_data);` This function should be updated to be able to call the new functions above. In both new cases, it should prompt the user for a frequency value which should be stored in register 0 of the `ApplicationData` to be used inside the new functions. The prompts and error outputs should match the example executions shown above.

Additional Documentation

- sin
- floor
- %, modulo/remainder

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)

- Make a `square_fill_audio_track` function that fills the audio track with a square wave.
  - Square Wave
- Make a `triangle_fill_audio_track` function that fills the audio track with a triangle wave.
  - Triangle Wave
- Make a `pulse_fill_audio_track` function that fills the audio track with a pulse wave. This should prompt the user for an additional value of a `duty_cycle`. A `duty_cycle` value of `0.5` should produce a pulse wave identical to your square wave.
  - Pulse Wave